TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

LINGYU ZHU

TEACHING DEVELOPMENT PROJECT: GENE EXPRESSION PREDICTION WITH DEEP LEARNING

Master of Science thesis

Examiner: University lecturer Heikki Huttunen
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 1 February 2017

ABSTRACT

LINGYU ZHU: TEACHING DEVELOPMENT PROJECT: GENE EXPRESSION PREDICTION WITH DEEP LEARNING
Tampere University of Technology
Master of Science thesis, 59 pages
May 2017
Master's Degree Programme in Information Technology
Major: Data Engineering
Minor: Signal Processing
Examiner: University lecturer Heikki Huttunen
Keywords: Gene expression, histone modification, deep Learning, convolutional recurrent neural network

Histone modifications are playing an important role in affecting gene regulation. In this thesis, a Convolutional Recurrent Neural Network is proposed and applied to predict gene expression levels from histone modification signals. Its two simplified variants: Convolutional Neural Network and Recurrent Neural Network, and one state-of-the-art baseline: DeepChrome are also discussed in this work. Their performances are evaluated with gene expression data that derived from Roadmap Epigenomics Mapping Consortium database by the Receiver Operating Characteristic, Area Under the Curve and statistical analysis. As a result, the Convolutional Recurrent Neural Network model achieves the best performance compared to the other models.

For teaching development of a pattern recognition and machine learning course from Tampere University of Technology, an approach of integrating theory and practice is used. Video recording, weekly exercises and competition are worked as the auxiliary parts of lectures, which helps the students have a better understanding of the theoretical knowledge and learn how to solve different kind of practical problems. We used histone modification data for a competition on this course, and this competition would be discussed emphatically in this thesis. For the competition, we motivated the students to develop multiple machine learning algorithms to accurately predict gene expression levels on five core histone modification masks. During the period of this course, the competition *Gene Expression Prediction* received 888 entries that submitted by 105 teams with 184 players. This thesis includes the analysis and summary of the outcomes from the competition. Additionally, the learning assessment is also discussed in this thesis.

PREFACE

With the rapid development of biotechnology and the advent of microarrays, there is an enormous explosion of biological data. This massive data has made the conventional analysis methods face a tremendous challenge. Different machine learning methods have draw attention from people due to their state-of-the-art performance in this field, especially deep learning. Our goal in this work is to predict gene expression levels with deep learning, and we chose *Gene Expression Prediction* as the topic of competition which is a practical auxiliary extension of a pattern recognition and machine learning course from Tampere University of Technology. This competition was jointly organized by Tampere University of Technology and University of Tampere. Most of the participants are the students from those two universities, they have different backgrounds: information technology and computing biology. The students could obtain both of the theoretical knowledge and practical skills from this course if they attend all of the "activities" that this course involved. The analysis between the design of the course from faculty and the feedback from the competition participants makes an excellent closed loop of reference for the future development of this course.

I would like to express my deep gratitude to my supervisor Heikki Huttunen for excellent guidance, reviewing the manuscript, and giving constructive comments of my thesis during this project. I also warmly thank Professor Matti Nykter and Juha Kesseli who have always being willing to providing support and insightful suggestions whenever needed. Moreover, I wish to thank all my coworkers for their contribution to this *Gene Expression Prediction* competition, and my warm thanks are extended to all the competition participants for their enthusiasm and invaluable feedback.

Finally, I would like to appreciate the valuable encouragement and support from my kind parents and all warm friends.

Lingyu Zhu

22 May, 2017

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADAM	Adaptive Moment Estimation
AUC	Area Under the Curve
BPTT	Back Propagation Through Time
CART	Classification and Regression Trees
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GRU	Gated Recurrent Unit
KNN	K-nearest Neighbors
LR	Linear Regression
LSTM	Long Short Term Memory
REMC	Roadmap Epigenomics Mapping Consortium
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TanH	Hyperbolic Tangent
TN	True Negative
TP	True Positive
TPR	False Positive Rate
TSS	Transcription Start Site
TTS	Transcription Termination Site

1. INTRODUCTION

Gene expression prediction can be regulated from various gene activity stages that range from the transcription of DNA-RNA to multiple post-translational protein modifications. The wide variety of mechanisms probably will facilitate or suppress the gene expression regulation. Histone modification is one of the most essential factors that affect gene expression from a transcription aspect. Histone proteins serve to wrap the DNA strands into chromosomes and select the genes that would get transcribed in cells. [1], [2].

The gene expression levels can be modeled and predicted using different machine learning algorithms. Machine learning is a data analysis method that enables computers to have the ability of model building and learning automatically. It helps computers to do pattern recognition and to find hidden insights without being programmed explicitly. In comparison to the conventional biological computational methods, machine learning has been put into practice with the research on studying the correlation of gene expression and histone modifications in many attempts [3]. For instance, quantitative models [4], Support Vector Machine (SVM) model [5] and Random Forest (RF) Classifier model [6] have been proposed to correlate the gene regulation with histone modification signals. As the averaging methods that previous studies used for capturing the input features from gene transcription region leave out some important details [4], and predicting separately for different feature regions using multiple models will seriously affect the final prediction accuracy. To avoid these problems, a deep learning method on this topic was proposed recently in [7].

Deep learning is a branch of machine learning with more complicated algorithms which can model features with high level abstraction from data. It has achieved the state-of-the-art performance in several fields such as image classification [8], [9], [10], semantic segmentation [11] and speech recognition [12], [13]. Recently, deep learning methods have also achieved success in computational biology [7], [14], [15]. Therefore, different machine learning methods, especially deep learning, have started being applied on this work for predicting gene expression levels from five core histone modifications [16]. In this thesis, we study how the Convolutional Recurrent Neural

Network (CRNN) performs for predicting the gene regulation tasks and compare its performance to Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and a convolutional baseline: DeepChrome [7]. Additionally, this work was accepted as a conference paper [17] on 11 April, 2017.

We used the gene activity prediction as an example case for a teaching experiment, and one machine learning competition *Gene Expression Prediction* was jointly organized by Tampere University of Technology and University of Tampere. Its outcome and feedback from the participants will also be discussed in this thesis.

The remainder of this thesis is arranged as follows. In Chapter 2, the theoretical background is presented, it includes machine learning, neural networks, deep learning backgrounds and the existing computational methods of predicting gene expression levels. Chapter 3 introduces data and the architecture of the proposed models. In Chapter 4, the performance of the studied models is evaluated. Additionally, the result of the *Gene Expression Prediction* competition and learning assessment is also presented in Chapter 4. Finally, conclusions and possible future work are discussed in Chapter 5.

2. THEORETICAL BACKGROUND

2.1 Machine Learning

The definition of machine learning: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" that proposed by Tom Mitchell in [18] is widely quoted. The working principle of machine learning is building a data driven model which can produce reliable predictions based on the computations in the training process. It can be applied in many fields, for example, financial services, health care, marketing and sales, transportation and so on. In this thesis, it is a binary classification task with biological genomic data. The ultimate goal is to learn and select a hypothesis h, which minimizes the differences between predictions $\hat{\boldsymbol{y}}_i = (\hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{in})$ and desired labels $\boldsymbol{y}_i = (y_{i1}, y_{i2}, \ldots, y_{in})$ of input $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im})$, from a class of hypotheses $\mathcal{H}$ and accurate predict gene expression levels. The i corresponds to the i-th instance that ranges from 1 to N, where N is the number of total instances. The m and n represent the input and output dimensions of each instance, respectively. Moreover, machine learning algorithms are typically distinguished into the following three categories: supervised learning, unsupervised learning and semi-supervised learning, depending on the data available and the purpose of different tasks.

Supervised learning: Supervised learning is where the target labels are known and you try to figure out one hypothesis h from hypotheses set $\mathcal{H}$ that can map the input $\boldsymbol{x}_i$ to output $\boldsymbol{y}_i$, which is illustrated in Figure 2.1. The supervised learning algorithms iteratively make predictions on labeled examples and are corrected by minimizing the errors between the predictions $\hat{\boldsymbol{y}}_i$ and the known desired labels $\boldsymbol{y}_i$. Supervised learning can be further grouped into two subcategories: regression and classification. The regression is used to model the correlation between input samples and targets which are continuous while the targets of classification are categorical. Depending on the different tasks, various supervised learning algorithms can be applied, such as Linear Regression (LR), K-nearest Neighbors (KNN), SVM, decision trees, RF and so on. [19], [20]

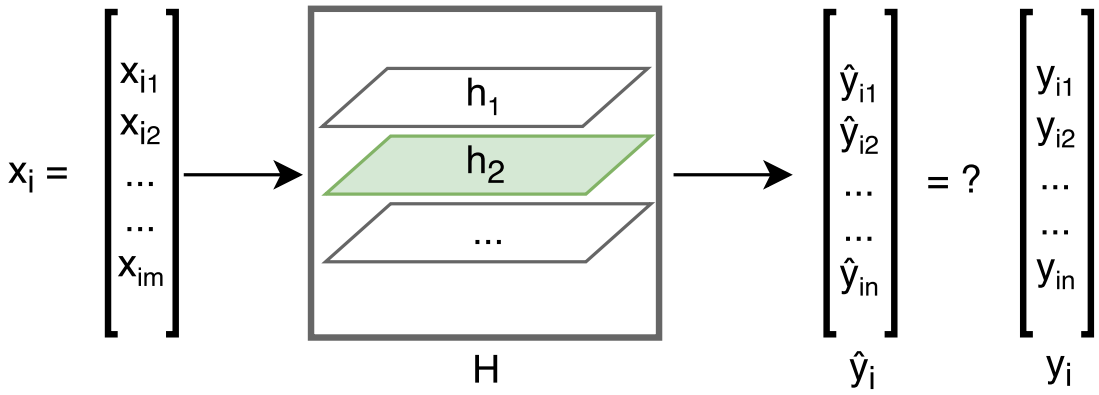

Figure 2.1 *Mapping input to output.*

Unsupervised learning: Unsupervised learning refers to the cases that the data has no target labels. The goal of the unsupervised learning algorithms is to find out the hidden structure within the data. They are called unsupervised learning because the "right answer" is unknown and there is no "supervisor" here for supervising the learning process. The unsupervised learning can be also grouped into two subcategories: clustering and association. Clustering, such as k-means [21] and self-organizing maps [22], [23], helps us to discover the inherent property within the data while the association rule learning algorithms, like Apriori [24], discover the rules that represent the large portion of data.

Semi-supervised learning: Semi-supervised learning sits in between supervised learning and unsupervised learning, it involves the function estimation on labeled and unlabeled data. The proposal of this modeling method is motivated by the fact that the cost associated with labeling data is usually high, whereas the unlabeled data is relatively cheap and easy to collect. Semi-supervised learning uses both labeled and unlabeled data for training, typically a large amount of unlabeled data with a small amount of labeled data. A mixture of supervised and unsupervised learning techniques can be used in this concept. [25], [26]

As the task in this thesis is a supervised learning problem, few machine learning algorithms related to supervised learning such as LR, KNN, SVM, RF, and XGBoost are introduced in this section.

2.1.1 Linear Regression

LR is the very basic predictive analysis for regression problems. Several linear regression analyses are commonly studied such as simple linear regression, logistic regression, multiple linear regression, ordinal regression, multinominal regression

and discriminant analysis. For visualization purpose, the concept and basic principle of simple linear regression is particularly discussed.

As the linear regression, in this case, concerns only the study of one independent variable, people named it as simple linear regression. It is an elementary and statistical method that enables us to figure out the statistical relationships between two quantitative variables. One variable is usually called the explanatory or independent variable and refereed as x_i here. Another variable, referred as y_i, is regarded as the response or dependent variable. In simple linear regression, we try to form and draw a straight line with the prediction $\hat{y}_i$ which are mapped as:

$$\hat{y}_i = ax_i + b \tag{2.1}$$

where the parameters a and b are constants, and represent the slope and intercept of the final optimal regression line, respectively. i is the i-th instance of total N samples. The goal is to find the best fitting straight line, usually called regression line, through all data points.

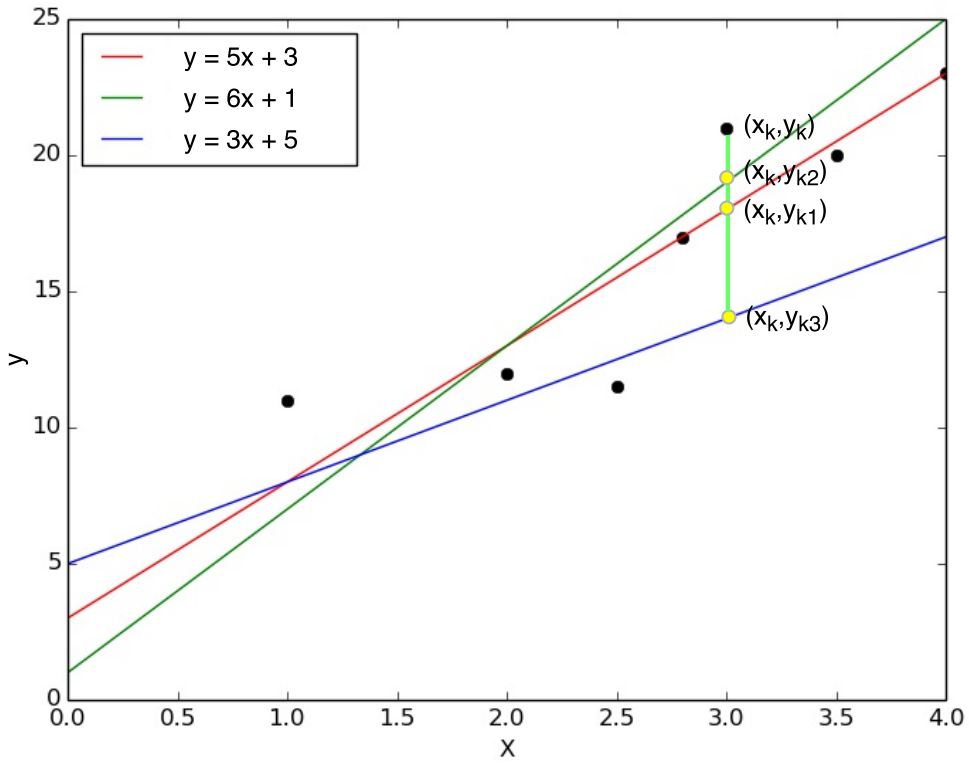

Figure 2.2 *Simple linear regression solutions.*

An example figure with several prediction solutions is shown in Figure 2.2. Figure

2.2 consists of six black points and three straight lines. The six black dots have their corresponding independent variable values x_i and desired dependent variable values y_i. The three straight lines consist of the predicted values $\hat{y}_i$ for each possible value of x_i. The three dots that are marked as yellow represent the predicted values on one same x_k, where the k means the k-th instance. The top black point corresponds to the desired dependent variable value y_k on this x_k. The distance between each yellow dot and the black one on the same x_k is called as prediction error. As we can see, the yellow point (x_k, y_{k2}) on the green line is near the black point (x_k, y_k) most. By contrast, the distance between the yellow point (x_k, y_{k3}) on the blue straight line and that black dot is much larger than the others and therefore its prediction error of this "blue solution" is the largest one on this independent variable value x_k. By summing the squared prediction errors of all data points with each of the three straight lines in Figure 2.2, we conclude that the red line is the best fitting line in this case as it has the minimum squared errors. The best fitting line is defined as the regression line that minimizes the sum of the squared errors of prediction. Accordingly, we could obtain the parameters a and b which minimize the total errors E in Equation 2.2 and Equation 2.3:

$$E = \sum_{i=1}^{N} (y_i - \hat{y}_i) \tag{2.2}$$

$$E = \sum_{i=1}^{N} (y_i - (ax_i + b)) \tag{2.3}$$

2.1.2 K-nearest Neighbors

KNN is a widely and commonly used classification technique because of its easy interpretation and low calculation time, and it is also often applied as a benchmark for the other models. KNN belongs to one member of the family of lazy learning, instance-based and competitive learning algorithms. Lazy learning is a learning method that only builds a model completely until the time when a prediction is required. In other words, the KNN algorithm is such a lazy learning method only does work at the last second. KNN algorithm also belongs to the instance-based algorithms as it models problems with all the training observations in order to make an accurate predictive decision. Moreover, as a competitive learning algorithm, KNN internally involves competition between data instances of model in order to make a predictive decision. The measurement of objective similarity, such as majority vote, among the data instances causes each data sample to compete to "win" and contribute to a prediction.

KNN is based on the idea of predicting unknown samples by comparing them with the most similar known values. Let us assume that we have two different types of data points in Figure 2.3. A spread of red circles and green squares belong to different classes. Our task is to find out which category the blue triangle belongs to. In the classification setting of KNN, the K indicates the number of nearest neighbors that we will consider when giving a new unseen data. The KNN algorithm will form a majority vote among the K most similar neighbors to this given unseen observation (x_{01}, x_{02}). The similarity is defined based on the distance metric among the two sets of data points. A common way to calculate the distance is the Euclidean distance which is formulated as:

$$D_{Euclidean} = \sqrt{(x_1 - x_{01})^2 + (x_2 - x_{02})^2} \tag{2.4}$$

where (x_1, x_2) corresponds to the coordinates of the points in Figure 2.3 and the (x_{01}, x_{02}) represents the unseen data point that needs prediction. Additionally, the Chebyshev and Manhattan distance are also often considered for calculating the distance metric [27]. Chebyshev distance is defined as the largest distance along all the coordinate dimension of two vectors. Manhattan distance sum the distances between two points measured along all axes. They are defined as below:

$$D_{Chebyshev} = \max(|x_1 - x_{01}|, |x_2 - x_{02}|) \tag{2.5}$$

$$D_{Manhattan} = |x_1 - x_{01}| + |x_2 - x_{02}| \tag{2.6}$$

The KNN runs through the whole dataset and computes the distance D between this unseen observation (x_{01}, x_{02}) and each training data. If the given positive integer K is 3, a 3 nearest neighbors area is shown in Figure 2.3. KNN then takes a majority vote among the points within this area. As we can see, among the three closest points to this blue triangle, two of them are the red circles and one is the green square. It is obvious that the observed blue triangle belongs to the red circle class with the $K = 3$. However, for the 7 nearest neighbors when the $K = 7$, 4 of them are the green squares which are more than the number of red circles. Therefore, this blue triangle gets assigned to the class with the largest probability $(\frac{4}{7})$. The choice of the parameter K is very crucial in this algorithm. We need to conclude an optimal K in order to get the best possible fit for the dataset. A higher K averages more voters in each prediction and hence is more resilient to outliers. Larger values of K will give smoother decision boundaries which means lower variance but increased bias. A small K gives a more flexible fit with higher variance but low bias. Besides, the value of K that we choose is usually odd to prevent the tie situations in a binary classification case.

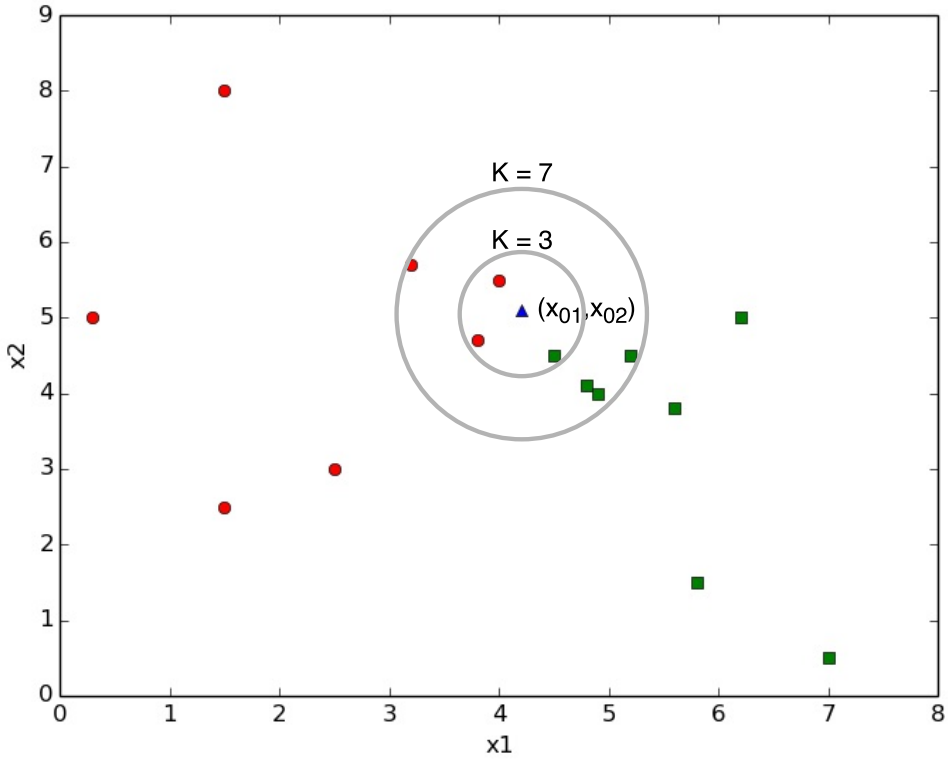

Figure 2.3 *KNN of the unseen observation.*

2.1.3 Support Vector Machine

SVM is a supervised machine learning algorithm which can be applied for both the regression and classification challenges, and it is mostly used for classification tasks. When solving a classification problem, the SVM is a discriminative classifier which will output an optimal hyperplane. This hyperplane is capable of categorizing new examples well.

In this example, a linearly separable set of 2 dimensional points which belong to one of two classes are discussed. The 2 dimension here corresponds to the number of attributes that locate their values on a particular coordinate. If we had three features, we could have a 3 dimensional graph. The goal is to figure out an optimal separating straight line in this example. We deal with the data points and hypothetical lines in the Cartesian plane instead of hyperplanes in a high dimensional space for problem visualization and simplification. However, the same working concept can be applied to classify data points in a higher dimension space.

In Figure 2.4, there exists several possible solutions to the problem. The operation of the SVM algorithm is aim to find the best separating hyperplane that mostly widen

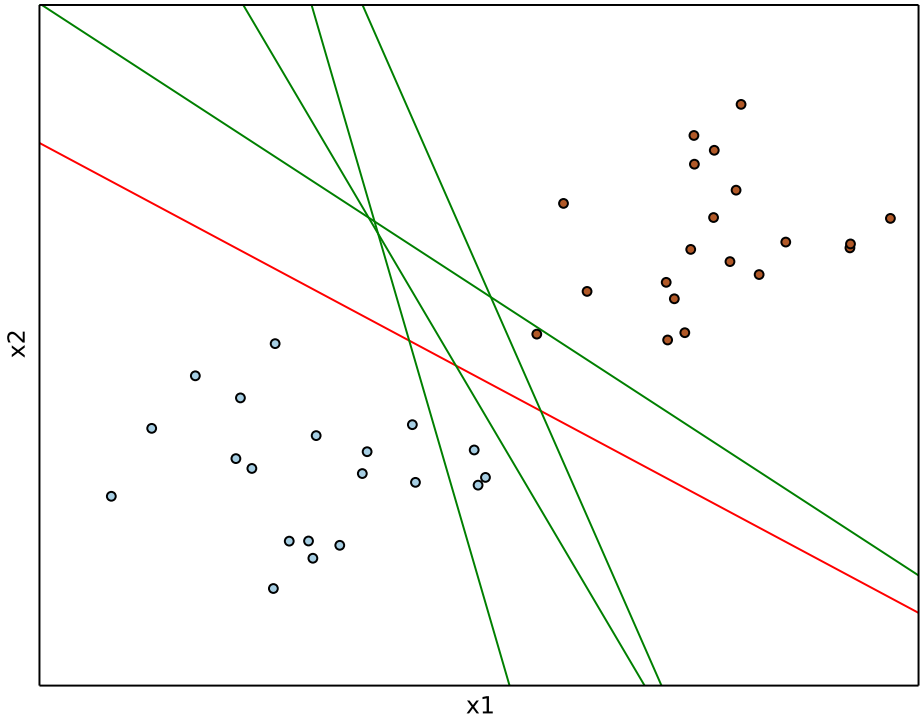

Figure 2.4 *Multiple solutions of SVM.*

the margin of the training dataset. The training process involves the minimization of the error function:

$$E = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^{N}\xi_i \tag{2.7}$$

subject to the constraints:

$$\boldsymbol{y}_i\Big(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b\Big) \geq 1 - \xi_i \tag{2.8}$$

where $\boldsymbol{w}$ is the coefficients vector, b is a bias, C is the capacity constant, and the larger the C the larger error is penalized. $\xi_i \geq 0$ represents the parameters of dealing with the non-separable inputs. $\boldsymbol{x}_i$ symbolizes the training examples while the $\boldsymbol{y}_i(+1, -1)$ represents the class labels. The kernel ϕ represents the operation that used to transform data points from the input to the feature space. [28] Finally, the problem of minimizing the error function E is equivalent to the maximization of margins. The red straight line in Figure 2.4 is the optimal decision boundary, among the available choices, that classifies the classes accurately and maximizing the margins also. This separating hyperplane and the margins are illustrated in Figure 2.5. In general, the samples that are closest to the hyperplane are called

support vectors. The support vectors are marked with yellow circles and plotted around the black dash lines (margins) in Figure 2.5.

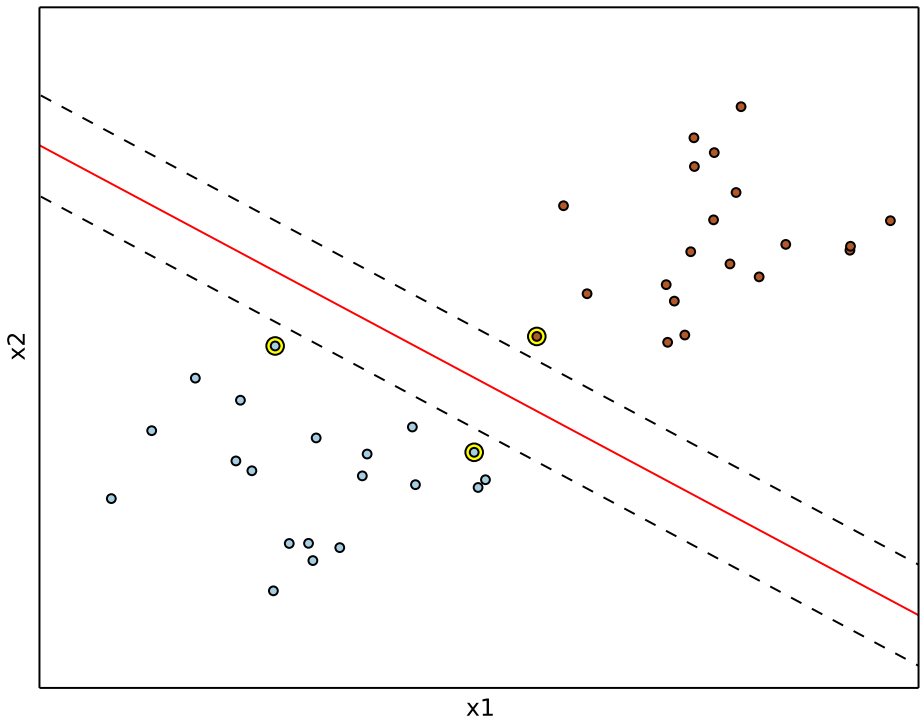

Figure 2.5 *Decision boundary of SVM.*

It is easy to visualize and have a linear hyperplane for the 2 dimensional data in this example. However, not every dataset is linearly separable. SVM has a technique called the *kernel trick* which can transform the feature vector $\boldsymbol{x}_i$ with the nonlinear mapping operation ϕ in Equation 2.8 into a higher dimensional space and convert the not separable problem to separable one. [28]

2.1.4 Random Forest

RF is based on a bagging technique with the idea that the combination of learning models can improve the classification accuracy significantly. Bagging averages the unbiased and noisy models and creates a new model with a lower variance. RF works as a large collection of uncorrelated decision trees and uses them to make a classification. In a decision tree, instances are entered at the roof and as they traverse down the tree the instances get bucketed into smaller and smaller subsets.

A decision tree is a straightforward divide-and-conquer diagram that combines individual attributes into a decision. However, the decision tree is easy to over-learn the data, which means that the created model will memorize the training set with a poor ability to generalize. RF overcomes this by training different "imperfect" trees. [29]

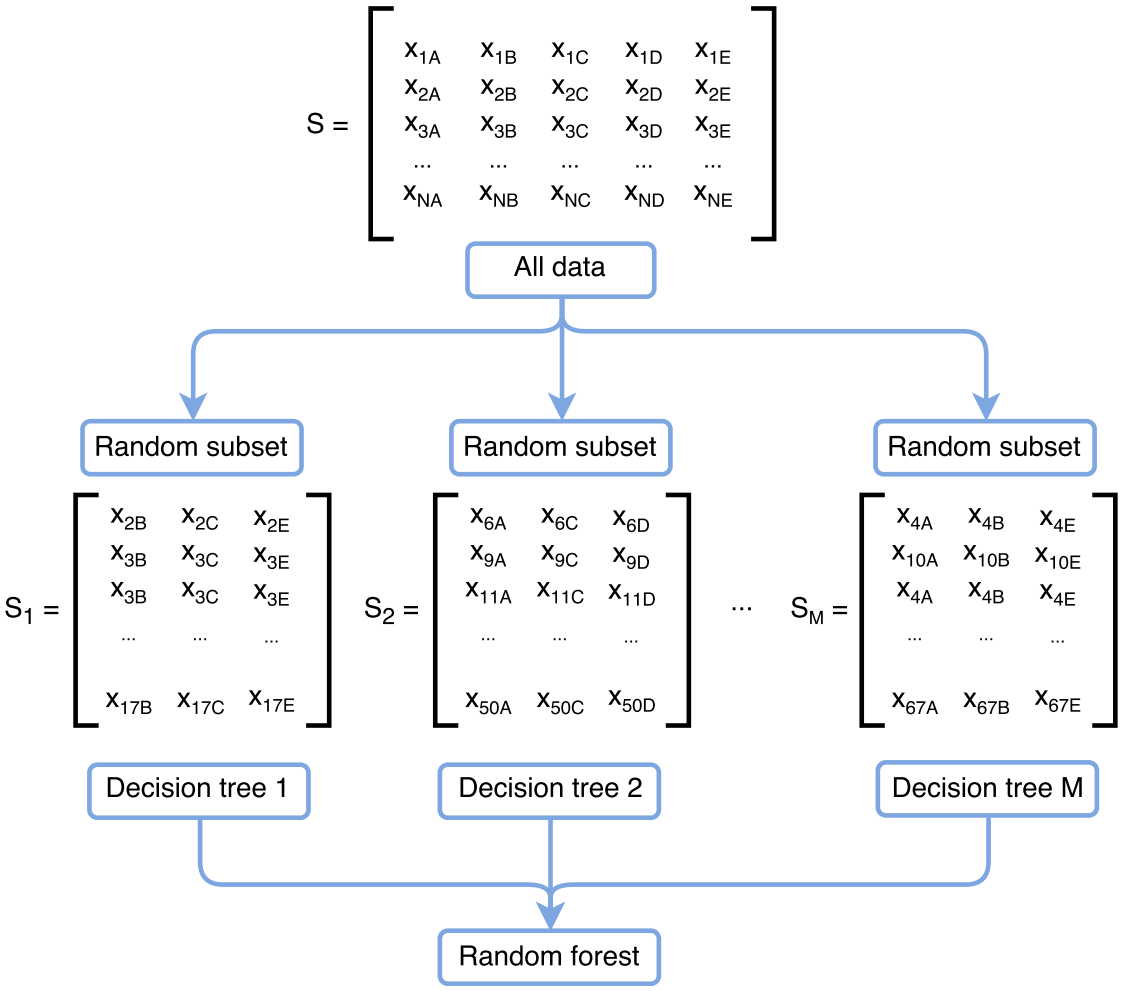

Figure 2.6 *Random forest.*

We suppose the matrix $\boldsymbol{S}$ in Figure 2.6 is the training set (N instances with 5 features). Each row corresponds to one instance and the columns are features. For example, the x_{1B} is feature B of the first sample, and x_{NE} represents the feature E of the N-th instance. Firstly, we create random subsets $\boldsymbol{S}_i$ from the matrix $\boldsymbol{S}$. Emphatically, the selection of rows is with replacement while the columns are sampled without replacement. Afterwards, we train separate decision trees based on each random subset, which is illustrated in Figure 2.6. Each decision tree executes the different variations of the main classification task. All these decision trees constitute a forest, and create a ranking of classifiers. The RF will make the prediction by taking the majority vote among the decisions from decision trees. Since the RF model

is created through dense randomness, hence it has good generalization and is robust against over-fitting. RF also includes the property of assessing feature importances by randomly shuffling each feature and testing the accuracy together.

2.1.5 XGBoost

XGBoost stands for the eXtreme Gradient Boosting. It is an implementation of gradient boosting machine which was proposed by Friedman in 2001 [30]. XGBoost is created for solving supervised learning challenges, where we map the training dataset $\boldsymbol{x}_i$ with multiple features to target variable $\boldsymbol{y}_i$.

Tree ensemble is a model of XGBoost and is also a set of Classification and Regression Trees (CART). An example of classification tasks on whether people like computer games or not was discussed in [31]. The authors classified the "candidates" into different leaves, and assigned real scores such as $+2.0, +0.1, -0.1$ to each corresponding leaf. A real score value is associated with each leaf of CART, and this is where the CART different from decision trees which just contain decision values. An example of a tree ensemble model with two trees is shown in Figure 2.7 and Figure 2.8. The prediction scores of each individual tree are summed up to make the final score.

When the input $\boldsymbol{x}$ is given, the tree ensemble model predicts the output $\hat{\boldsymbol{y}}$ as:

$$\hat{\boldsymbol{y}}_i = \sum_{k=1}^{K} f_k(\boldsymbol{x}_i), \quad f_k \in \mathcal{F} \tag{2.9}$$

where the $\mathcal{F}$ is the space of CART, f_k corresponds to one independent tree and the K is the total number of trees from the ensemble model. For the feasibility of optimizing regularized objective function, the additive strategy is adopted in the learning process, and the objective function at t-th iteration is defined as $\mathcal{L}^t$:

$$\mathcal{L}^t = \sum_i \ell\Big(\boldsymbol{y}_i,\ \hat{\boldsymbol{y}}_i^{t-1} + f_t(\boldsymbol{x}_i)\Big) + \Omega(f_t) \tag{2.10}$$

where the ℓ in Equation 2.10 is the differentiable loss function that calculates the target and prediction differences. The $\boldsymbol{y}_i$ is the desired target of i-th instance while $\hat{\boldsymbol{y}}_i^{t-1}$ represents the predictions of the i-th instance at $(t-1)$th iteration, and the second term $\Omega(f_t)$ is the penalty. One new tree would be added when the learning is fixed based on what we have learned. If the first and second derivatives of loss ℓ are defined as g_i and h_i:

$$g_i = \partial_{\hat{\boldsymbol{y}}_i^{t-1}} \ell(\boldsymbol{y}_i,\ \hat{\boldsymbol{y}}_i^{t-1}) \tag{2.11}$$

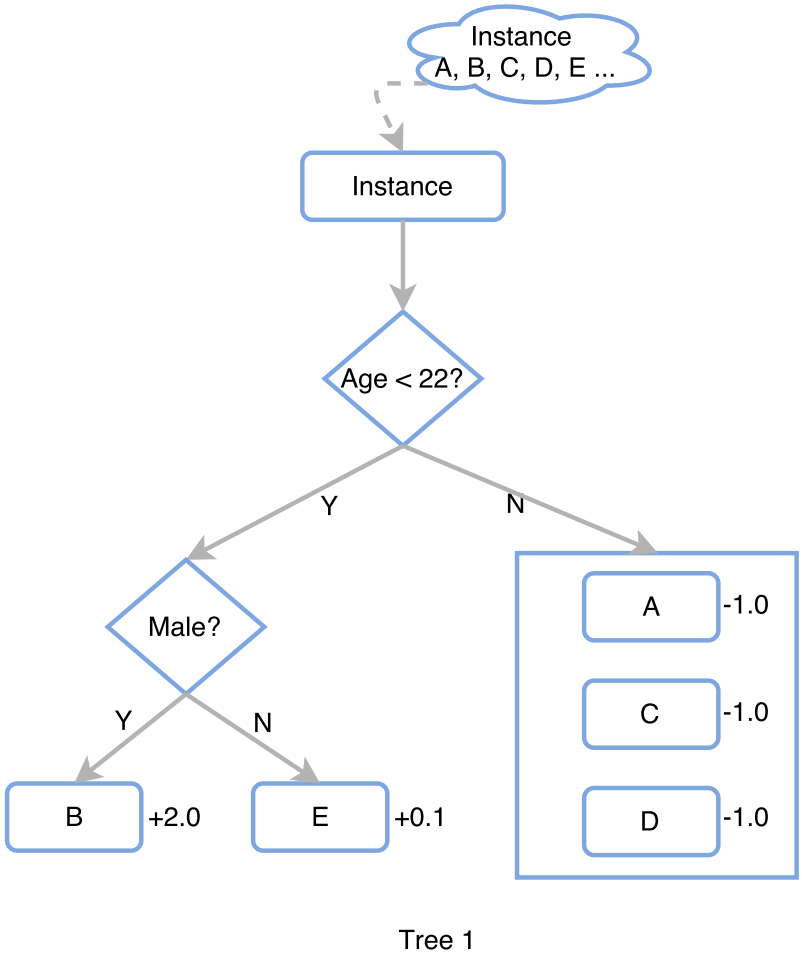

Figure 2.7 *Tree ensemble model (age and male), adapted from [31].*

$$h_i = \partial^2_{\hat{\boldsymbol{y}}_i^{t-1}} \ell(\boldsymbol{y}_i,\ \hat{\boldsymbol{y}}_i^{t-1}) \tag{2.12}$$

a simplified objective at iteration t is obtained as:

$$\mathcal{L}^t = \sum_i [g_i f_t(\boldsymbol{x}_i) + \frac{1}{2} h_i f_t^2(\boldsymbol{x}_i)] + \Omega(f_t) \tag{2.13}$$

The tree structure quality measurement can be managed by summing the statistics g_i and h_i of each leaf. However, it is impractical to enumerate all the tree structures available. Thus, an algorithm of starting from a single leaf and increasing branches iteratively to the tree is applied instead for generalization. [31]

XGBoost is a scalable distributed gradient boosting system that designed for its efficiency, flexibility and portability. This system has very high execution speed compared to the other existing popular implementations of gradient boosting and bagged decision trees. Its parallel and distributed computing enables quicker model exploration which makes learning faster. [32]

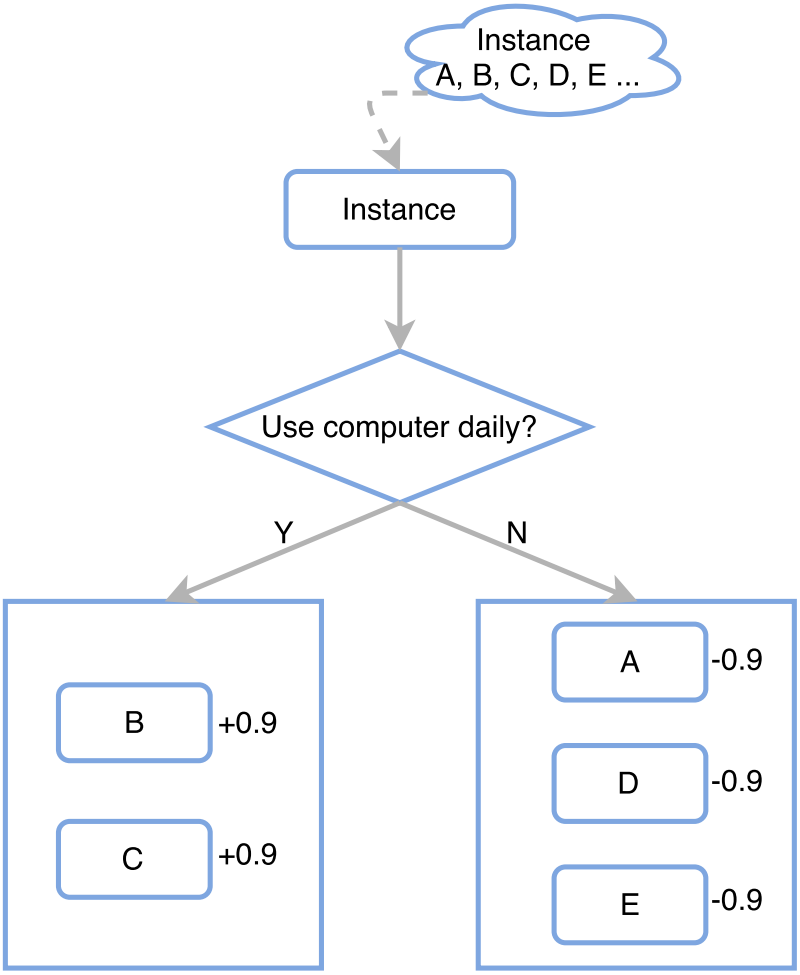

Figure 2.8 *Tree ensemble model (use computer daily), adapted from [31].*

2.2 Neural networks

2.2.1 Feedforward Neural Network

Neural networks are one group of algorithms used for machine learning that model the data using artificial neurons. Neural networks are inspired from biological nervous systems. Figure 2.9 illustrates a simple biological neuron network. As can be seen, dendrites bring input signals to neurons, and axons convey the information from the cell body of a neuron to another neuron across synapses. Artificial neural network is one application that applied in computer science based on this biological nervous system, and it is drawn in Figure 2.10. In this figure, there are four layers ℓ: one input layer (*in*), two hidden layers (*hidden*) and one output layer (*out*). The circles that are marked as blue are called bias units. The vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ in the leftmost input layer is the input signal that conveyed along the axons. It is moving forward by encoding with the learnable synaptic strength ${w^{\ell}}_{pq}$ and b^{ℓ}, where

${w^{\ell}}_{pq}$ denotes the weights associated with the connection between unit q in current layer ℓ and unit p in previous layer and b^{ℓ} represents the bias in current layer ℓ. The activation of each unit in the first hidden layer is obtained as below:

$$a_q^{hidden} = f(\sum_{p,q} w_{pq}^{hidden} \boldsymbol{x}_p^{in} + b^{in}) \tag{2.14}$$

The $\boldsymbol{x}_p^{in}$ in Equation 2.14 corresponds to the units in the input layer. The activation principle of the first hidden layer can be similarly applied into the successive hidden layers. Finally, the dendrites will bring the intermediate operations from the last hidden layer into the output layer where the output will be generated. This neural network defines the hypothesis as below for producing the output:

$$h_{w,b} = a_q^{out} = f(\sum_{p,q} w_{pq}^{out} a_p^{hidden} + b^{hidden}) \tag{2.15}$$

Generally, in each unit, the multiplications: $w_{pq}^{\ell} x_p^{\ell-1}$ or $w_{pq}^{\ell} a_p^{\ell-1}$ are get summed with bias $b^{\ell-1}$, and followed with an activation function f. For the activation function, it serves as a decision function which maps the original space into different partitions of the output. Various activation functions such as the hyperbolic tangent (TanH) and rectified linear unit (ReLU) activation functions, can be chosen depending on different implementation purpose. [33], [34]

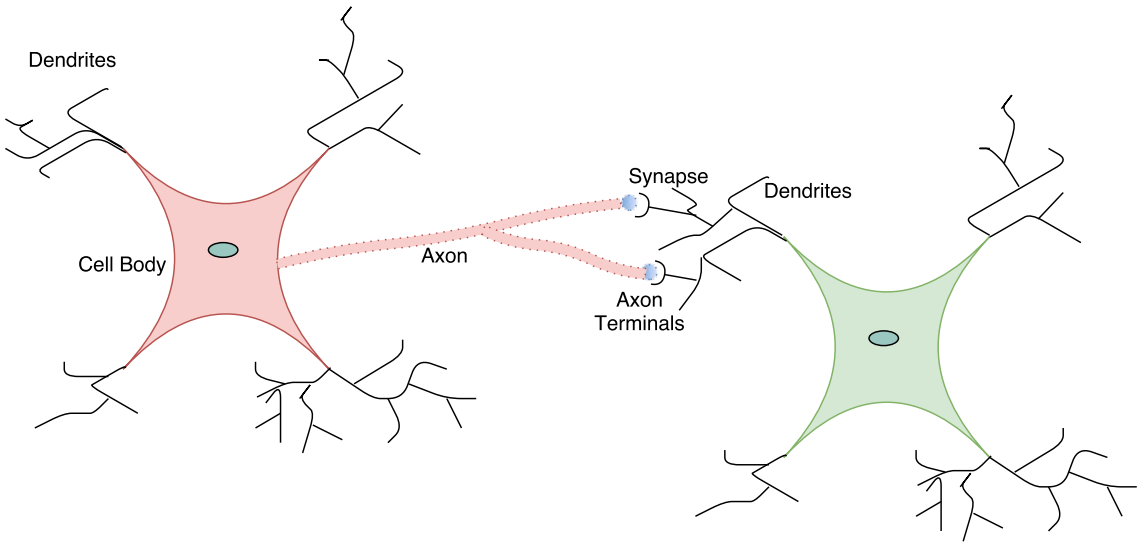

Figure 2.9 *A simple biological neuron network.*

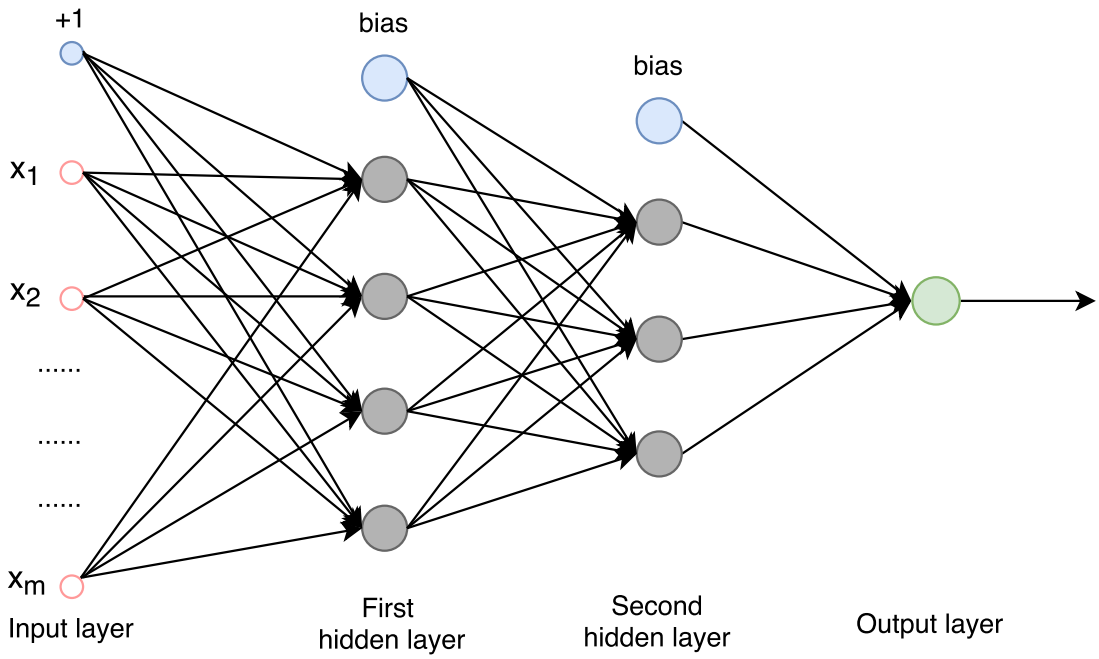

Figure 2.10 *A feedforward neural network model.*

2.2.2 Back propagation

Backward propagation is a common optimization method of training artificial neural networks. In [35], the good performance of back propagation applications in several neural networks was discussed, and its importance got widely known after that. Back propagation basically has two phases: propagation and weights update.

Propagation: When the neural network is fed with an input, it forward propagates the input through the whole network to output layer where the output is generated. Then the error between desired outputs $\boldsymbol{y}_i$ and predictions $\hat{\boldsymbol{y}}_i$ of i-th instance is computed from a loss function, which is denoted as $L(\boldsymbol{w}, d)$:

$$L(\boldsymbol{w}, d) = \sum_i ||\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i||^2 \tag{2.16}$$

where the $\boldsymbol{w}, d$ are the weights and biases that we discussed in Subsection 2.2.1. Then the network propagates the error backwards until the weights, connecting all the units between layers, have their corresponding associated error values which represent their contribution to the forward propagation process.

Weights updating: In this phase, based on these error values, backpropagation computes the partial derivative $\partial_{\boldsymbol{w}} L(\boldsymbol{w}, d)$ of the loss function $L(\boldsymbol{w}, d)$ with respect to the weights in the network. The gradient is used to minimize the loss function and get the optimization by updating the weights. This is not merely just a fast learning

algorithm but also an enlightening process for giving us the insight of how quickly the cost changes and how the weights updating changes the overall behavior of the network.

- Stochastic Gradient Descent

  Stochastic Gradient Descent (SGD) is one of the most popular and common algorithms to perform optimization on objective function $L(\boldsymbol{w}, d)$. It updates the weights in the opposite direction of the gradient of objective function $L(\boldsymbol{w}, d)$ which is denoted as:

  $$\boldsymbol{w} = \boldsymbol{w} - \eta \, \partial_{\boldsymbol{w}} L(\boldsymbol{w}, d) \tag{2.17}$$

  The learning rate η determines the step size of reaching a valley, and it is typically a small enough number which tends to give good convergence to a local minimum. SGD helps to find a better local optimum by performing frequent updates with a high variance, and to converge to the global minimum ultimately by keeping overshooting. Generally, SGD computes the weights updating using a few training examples or a minibatch from the training set rather than a single example, as this leads to a more stable convergence and efficient matrix operations.

- Adaptive Moment Estimation

  In this thesis, we use Adaptive Moment Estimation (ADAM) [36] as the optimizer of the learning process. It estimates the gradient mean (m_t) and element-wise squared gradient (v_t) by employing exponential moving average while adding bias correction. The biases β_1, β_2 corrected first and second order moments at time step t are computed as $\hat{m}_t$ and $\hat{v}_t$:

  $$\hat{m_t} = \frac{m_t}{1 - \beta_{1,t}}, \tag{2.18}$$

  $$\hat{v_t} = \frac{v_t}{1 - \beta_{2,t}} \tag{2.19}$$

  The parameters $\boldsymbol{w}_{t+1}$ at time step $t + 1$ are updated by following the ADAM updating rules as below:

  $$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{2.20}$$

  where η represents the learning rate, and the ϵ is a smoothing term that avoids the denominator being zero.

2.3 Deep learning

Deep learning, also widely known as deep structured learning, is a special branch of machine learning. It is distinguished from the neural networks with single hidden layer by its depth. Technically, the neural networks with more than one hidden layer are considered as deep learning. Each layer of deep learning networks can learn features with different abstraction and complexity. The more deeper you go into the deep networks, the more abstract features you can extract from it, this is known as the feature hierarchy.

In this section, the fundamental knowledge of few common remarkable deep neural networks, such as CNNs and RNNs, are presented successively.

2.3.1 Convolutional Neural Networks

Regular neural networks are made up of neurons with learnable weights and biases. The input layer receives a vector input and conveys it across hidden layers which are composed of neurons. The neurons in each layer are fully connected to the neurons in previous layer and next layer, and have no connections with the neurons in the same layer. Such a network structure does not take the spatial structure of the data into account if the input is some spatial one. CNNs perform very well for spatial data, and are able to encode the data properties into architecture.

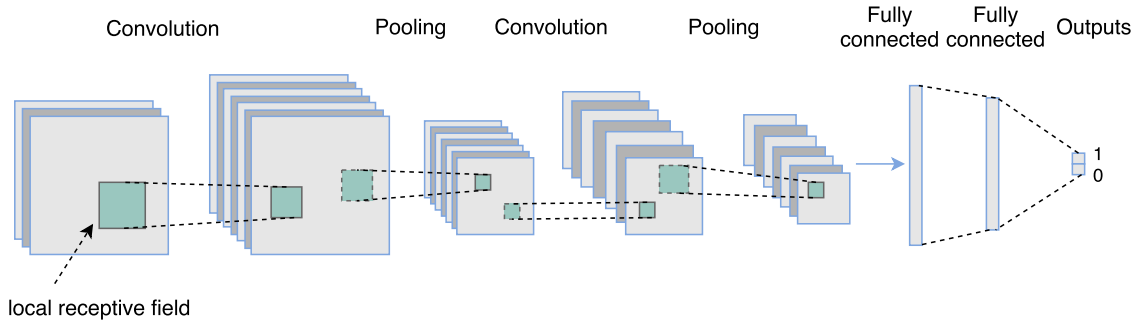

Figure 2.11 *Architecture of Convolutional Neural Network (adapted from [37]).*

Generally, the CNN is a sequence of layers, and each layer transforms the volume of activations to the subsequent layer. Basically, the CNN architecture, illustrated in Figure 2.11, has three main layer types which are stacked: convolutional layer, pooling layer and fully-connected layer. CNNs perceive images, assuming the spatial data is image here, as volumes in three dimensions: width, height and depth, which is shown in Figure 2.12. For the input layer, the width and height are measured

by the number of pixels along x and y coordinates while the depth here is referred as channels, which corresponding to three RGB color space, and illustrated along z coordinate. The input volume is further fed into the network by convolutions. In the convolutional layer, features abstraction is executed with dot product between filter weights and corresponding local input pixels, which is shown in the convolutional layers of Figure 2.11. A two dimensional activation map will be generated by sliding each filter across the width and height of the whole input volume. Usually non-linear activation function, such as ReLU, will be added after each convolutional layer in order to introduce non-linearity into the convolutional network. Depending on the number of filters used in the project, the output of this layer can be created by stacking the activation maps from different filters along the dimension: depth. With regard to reducing the workload of the network, the pooling layer, one down sampling operation, usually follows the convolutional layer and decreases the spatial size of its input. After several convolutional and pooling layers, the fully connected layers which fully connect the features with high-abstraction from previous layers come after and finally generate the network output.

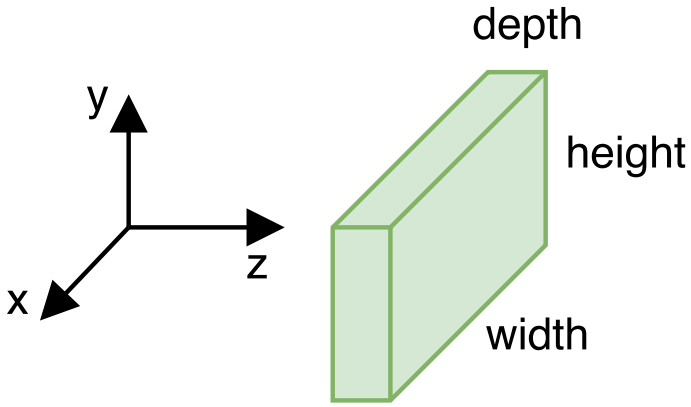

Figure 2.12 *Three dimensional volume.*

CNNs process the spatial and ordered data in an architectural way with three basic concepts: receptive fields, shared weights and pooling technique.

Receptive fields: In CNN, each unit in the hidden layer is only spatially connected to a local region of units in the previous layer, which is called the receptive field for the hidden neuron. Each connection between input pixel within the local receptive field and the hidden neuron learns a weight, and an overall bias is also learned by the hidden neuron. When sliding the local receptive field over the entire input image, there is a specific hidden neuron corresponding to each receptive field in each hidden layer. The size of the local receptive field and the number of hidden layers depends on the kernel size and numbers respectively. To illustrate this concisely in Figure

2.13, we assume the input is a gray scale image, and the first hidden layer is built up by sliding the local receptive field from the up left corner of the image. Then the local receptive field is slided right or down over the stride length as the model sets.

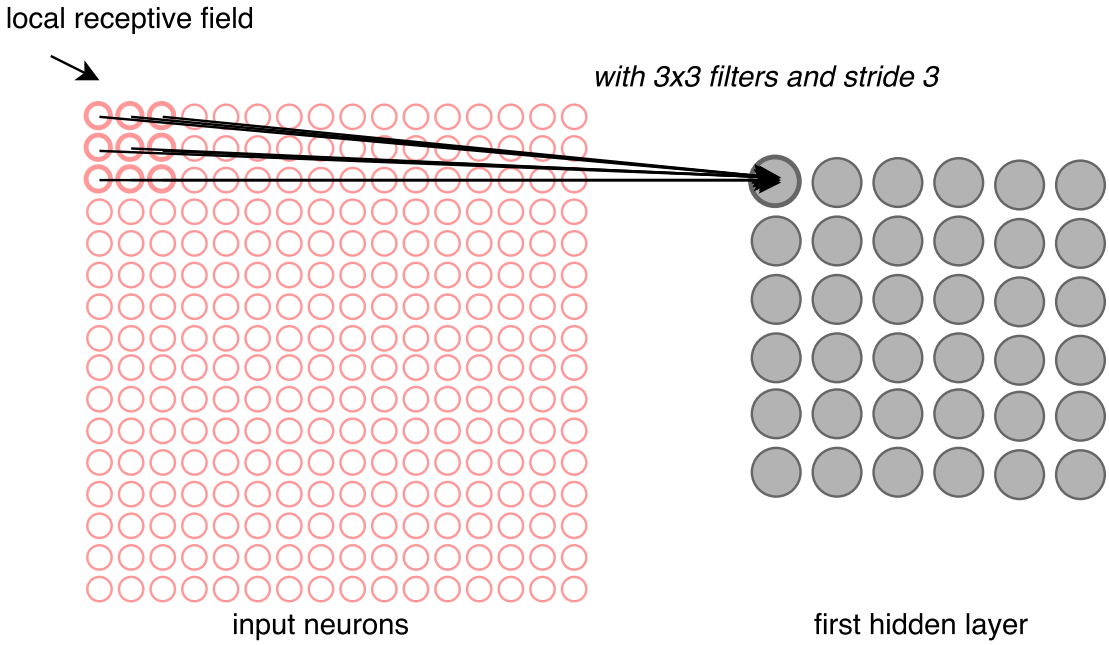

Figure 2.13 *Receptive fields (adapted from [38]).*

Shared weights: One principal advantage of CNNs is the shared weights of convolutional layers, which means that the same weights and biases are used when convolving one filter with different local receptive fields. The shared weights and biases achieve better generalization and highly improve the learning efficiency by reducing the number of parameters which involved in the convolutional layers. Besides, the shared weights also equip the model with the ability of detecting features regardless of their position in the visual field. A feature map is generated after the convolutional operation with this filter. The weights defining the feature maps are called the shared weights, and the biases are called the shared biases. The shared weights and biases define a filter.

Pooling: Pooling layers are typically applied after convolutional layers for down sampling and reducing the spatial size of the output from previous layer. Pooling operation efficiently reduces the amount of parameters and computation complexity of network. The most common pooling method is max-pooling. Basically, a pooling window, specifying local pooling areas, is to be set so that you do not need to pool over the entire matrix at one time. For example, the max-pooling of a 2×2 window is a process of traversing over the area within the pooling window and picking up the

largest element, which is illustrates in Figure 2.14. The output of the max-pooling layer is decreased into a half size matrix with a 2×2 pooling window of stride:2.

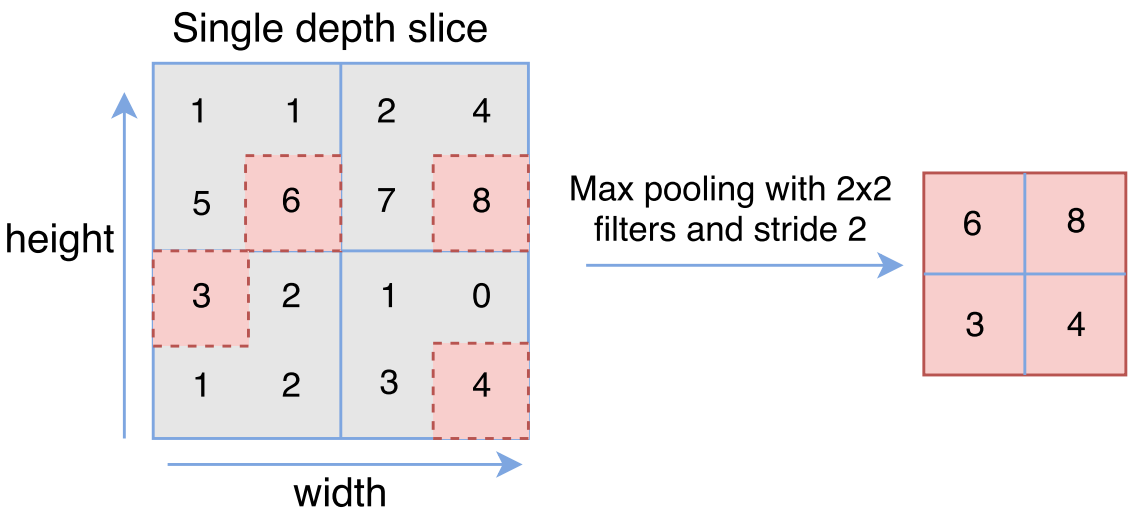

Figure 2.14 *Maxpooling (adapted from [34]).*

2.3.2 Recurrent Neural Networks

RNNs are a type of artificial neural network designed to recognize patterns in sequences of data, such as sound, written natural language, genomes and stock markets. RNNs were created in 1980's and started recently gaining popularity in network designs and powerful computation with graphic processing units. One principal difference between RNNs and the traditional neural network is the architecture of how neurons are connected to the others. Each neuron or unit is able to use internal memory to maintain information about the previous input. For example, when you read a message, you understand each word based on your perception of overall context as your memory has persistence.

The reason why RNNs are called *recurrent* is because RNNs execute the same task on each input element of sequence and produce output based on early computations. A typical recurrent network is shown in Figure 2.15. RNNs allow information to persist by including loops. If unfolding the loops in time, the RNNs can be regard as deep feed forward networks. The chain-like structure is shown in Figure 2.15. x_t, s_t and o_t are the input, hidden state and output at time step t, respectively. U, V are the parameters between input x_t and hidden state s_t, hidden state s_t and output o_t on current time step t. W is the parameters between successive hidden states. RNNs share the same parameters across each time step, and this significantly reduces the number of parameters in network. People usually call the s_t the memory

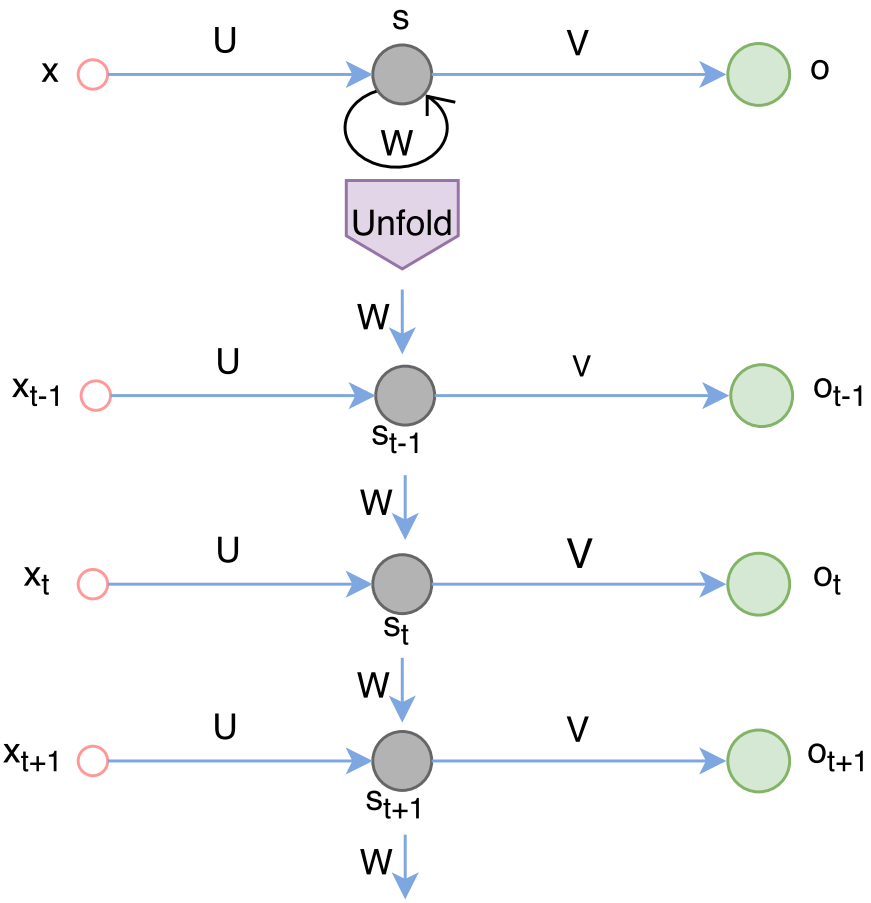

Figure 2.15 *A recurrent neural network with unfolding in time (adapted from [39]).*

of the network, as it captures the information from the previous time steps. s_t is calculated based on the input x_t at current time step t and previous hidden state s_{t-1}, which is presented as below:

$$s_t = f(Ux_t + Ws_{t-1}) \tag{2.21}$$

The network in Figure 2.15 has outputs at each time step, however, depending on different tasks the RNNs can be built differentiated by operating over sequences. Few examples are illustrated in Figure 2.16. The red circles represent the input vectors, gray ones save the state of RNNs and green stands for output vectors. The arrows indicate the information flowing directions. From 1 to 5, the first "one to one" is the vanilla model from fixed-sized input to fixed-sized output without RNN. The second "one to many" model is from fixed-sized input to the sequential output. "Many to one" models sequential input to the fixed-sized output. Furthermore, the final two "many to many" models work with the general and synchronized sequence input and sequence output. Compared to the Vanilla Neural Networks which have

the constraints of fixed-sized input, output and fixed amount of computational steps, the flexibility on operating over sequences of vectors makes RNNs remarkable.

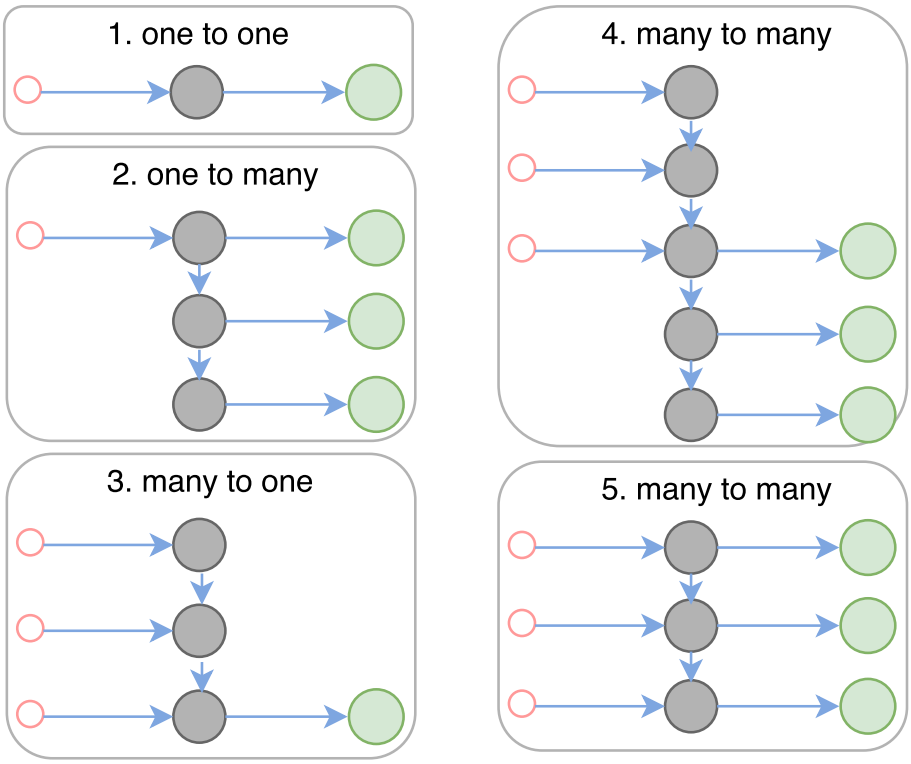

Figure 2.16 *Examples of Recurrent Neural Network sequential model (adapted from [40]).*

The purpose of RNNs is to classify sequential input accurately. It gets the error rate by comparing its predictions with the desired targets from test data. For minimizing this error rate, a gradient based technique called Back Propagation Through Time (BPTT) is applied. BPTT begins by unfolding the RNNs through time as shown in Figure 2.15. BPTT trains and back checks through the network in a sequential order. It significantly faster the process of training RNNs.

RNNs are capable of memorizing the information of arbitrary long sequences in theory, but in practice, s_t typically cannot capture very long-term dependencies because of the vanishing gradients problem. Two variants of RNNs: Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) are widely used today as the solutions to the vanishing gradients problem. The illustration of LSTM and GRU is shown in Figure 2.17 and discussed as below:

Long Short Term Memory: The LSTM was initially proposed by Hochreiter and Schmidhuber in 1997 [42]. Like all RNNs, a typical LSTM has a chain structure of repeating modules. Each module includes four interactive layers: input gate(i_t),

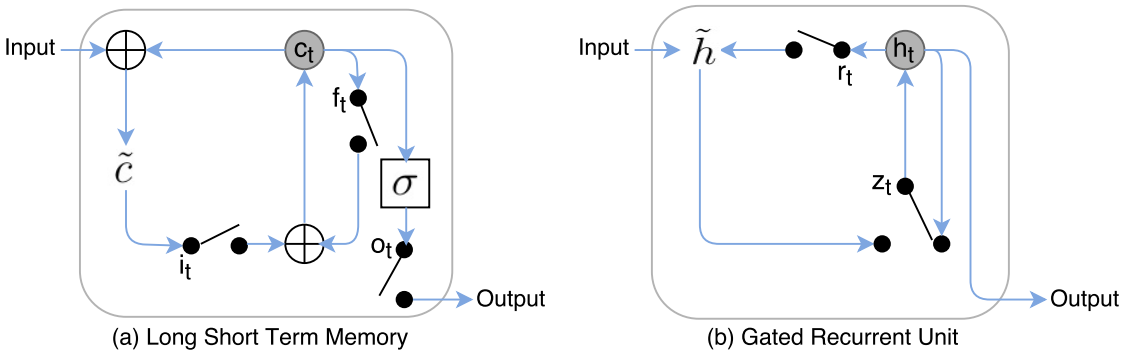

Figure 2.17 *Illustration of (a) LSTM and (b) GRU. (adapted from [41])*

output gate (o_t), forget gate (f_t) and cell state gate (c_t). The input gate and output gate regulate the input and output flow of the module. Forget gate layer decides whether to forget or reset the cell state. The cell state gate can be updated optionally by adding or removing information. If feeding sequential data $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ to the LSTM network, it will map the input into output by calculating each gate layer equations:

$$f_t = \sigma(W_f \cdot h_{t-1} + W_f \cdot x_t + b_f) \tag{2.22}$$

$$i_t = \sigma(W_i \cdot h_{t-1} + W_i \cdot x_t + b_i) \tag{2.23}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_c \cdot h_{t-1} + W_c \cdot x_t + b_c) \tag{2.24}$$

$$o_t = \sigma(W_o \cdot h_{t-1} + W_o \cdot x_t + b_o) \tag{2.25}$$

$$h_t = o_t \odot tanh(c_t) \tag{2.26}$$

where f, i, o and c are the forget gate, input gate, output gate and cell state gate respectively. t represents the current time step, and h_t is the signal that is going to be output from this module. σ denotes the logistic sigmoid function. All the W terms are the weights matrices corresponding to each gate of model, and the b terms are biases. $\odot$ is an element wise multiplication operation. LSTM unit enables the model to detect and capture the long distance dependencies by deciding whether to save the existing memory through introduced gates or not.

Gated Recurrent Unit: GRU is basically a variant of LSTM having no separate memory cells and output gate. As can be seen from Figure 2.17 (b), r and z represent the reset and update gates while h and $\tilde{h}$ are the activation and candidate activation operations. Unlike LSTM, the GRU exposes the whole memory state at each time step and balances the previous and new memory states with leaky integration. At time step t, the update gate z_t decides how much the previous h_{t-1} and new memory state h_t are to be forgotten or memorized, and is calculated based on current input i_t and previous memory state h_{t-1}. The candidate memory state is

computed by input and the reset gates r_t with previous memory state h_{t-1}, and this helps GRU to decide whether the detected features or previous states are necessary or not. [43] The updating and reseting mechanisms are listed as below:

$$h_t = (1 - z_t)h_{t-1} + z_t h_t \tag{2.27}$$

$$z_t = \sigma(W_z i_t + U_z h_{t-1}) \tag{2.28}$$

$$\tilde{h}_t = tanh(W i_t + r_t \odot U h_{t-1}) \tag{2.29}$$

$$r_t = \sigma(W_r i_t + U_r h_{t-1}) \tag{2.30}$$

2.4 Existing computational approaches for predicting gene regulation

Histone modifications are quantizable signals which flank on genes. Multiple research and biological technology advancement have capacitated us to profile the quantizable histone modification signals and to quantify the gene regulation more accurately and efficiently. Besides, various computational models such as LR, SVM, RF classifier and CNNs have been applied to predict gene expression levels with histone modification signals in last decade.

Research of the correlation between the gene regulation and histone modifications has been done in 2013 by Dong and Weng [3]. LR is a commonly used predictive machine learning model. Its purpose is to furthest minimize the summation of the squared errors of predictions and desired output to fit a straight line into dataset. Article [4] adopted a LR model on data that derived from human T-cell studies [44] to quantify the relation between the histone modifications and gene expression. It showed us that a small number of histone modification signals could make accurate predictions very close to the experiment of using all 38 histone modification marks. It also stressed that the relationship between the histone modification signals and gene regulation is general rather than cell type specific. Furthermore, [45] investigated a hybrid of LR algorithms and applied them to identify the gene regulation patterns by modeling histone modification signals. The authors indicated that the mixture of LR methods have a better performance than using a single LR model.

In 2011, a statistical approach of studying the relationship between gene expression and histone modifications was demonstrated in [5]. Its authors applied SVMs on worm dataset [46] to study the various aspects of gene expression by chromatin features. The regions flanking astride the Transcription Start Site (TSS) and Transcription Termination Site (TTS) were divided into bins with 100 basepairs and

were considered as the features of the SVM models. SVMs are the discriminative classifiers which try to find the hyperplane that gives the maximum margin of the labeled training data (supervised learning). Its results showed the quantitative relationship between the gene regulation and histone modification signals based on the incorporate informations by modeling different bins. Besides, it inferred the histone modificaton pairwise interactions with a separate LR model. Bayesian networks were also studied for modeling higher order correlation between the gene expression and histone modification marks.

Dong X built a quantitative model with RF classifiers to predict gene expression levels with histone modification signals in 2012. It performed feature selection by just keeping the bins which correlated with gene expression most, and studied the combinatorial analysis by grouping the 11 histone modifications into 4 functional categories. It proved that the gene expression status can be better predicted by jointing various groups of features. [6]

With the popularity of deep learning recently, deep convolutional networks were proposed to predict gene expression levels from histone modifications. In [7], a unified convolutional discriminative framework was developed. This article claimed that the developed DeepChrome model overcame the weaknesses of previous studies such as multiple models, combinatorial analysis and the failure of capturing subtle sequential differences of histone modification signals. Because the convolutional mechanism in DeepChrome allowed the automatic feature extraction along both the bins direction and histone signals direction. Its result has also shown that the DeepChrome outperformed the discussed early computational methods for predicting gene expression levels with 5 core histone modification signals on 56 cell types.

In conclusion, with the advancement of the biological techniques and development of computational applications of machine learning algorithms, especially the deep learning, the correlation between gene regulation and histone modifications could be modeled more accurately and efficiently. This topic is still a rather significative and promising issue as its improvement could support the medicine-health field greatly.

3. MATERIALS AND RESEARCH METHOD

3.1 Data Preparation

In this section, the materials and tool that used for generating and processing the data are introduced. Besides, the data extracting and preprocessing process is also discussed.

3.1.1 Materials

Histone modification is a post-translational modification process made on histone proteins which plays a fundamental role in impacting gene regulation [2]. The 5 core histone modification masks: H3K4me3, H3K4me1, H3K36me3, H3K9me3 and H3K27me3 [16], summarized in Table 3.1, and normalized RNA sequence data on 56 cell types are downloaded from Roadmap Epigenomics Mapping Consortium database (REMC) database [16].

Histone Marks	Associated with
H3K4me3	Promoter regions
H3K4me1	Enhancer regions
H3K36me3	Transcribed regions
H3K9me3	Heterochromatin regions
H3K27me3	Polycomb repression regions

Table 3.1 *Five core histone modification marks.*

GENCODE [47] supports genomics projects by providing updated human gene annotation. In order to find the gene coordinates in this work, the basic gene annotation file is downloaded from the latest version of GENCODE release 25 (mapped to GRCh37).

3.1.2 Bedtools

Bedtools is a powerful and flexible tool that allows various genome arithmetic on different genomics analysis tasks [48]. In this work, the *bedtools intersect* operation

on the genomic intervals from the 5 core histone modifications and basic gene annotation file is applied. It provides multiple parameters options for satisfying our different needs based on how we expect the intersections operations are reported. This work is done by screening for overlaps between two sets of genomic features using *bedtools intersect*.

3.1.3 Data extraction and preprocessing

Based on the gene coordinates that we get from the basic gene annotation file, the "start location" of the genes with genomic strand "+" is considered as the TSS, and the "end location" is considered as TSS with the genes of genomic strand "-". For keeping the subtle variations among the effects from histone modification signals on each gene, a 10,000 basepairs DNA region surrounding the TSS is chosen to be mapped into 100 consecutive bins, for which each contains 100 basepairs. [7] The feature window is constructed by intersecting the 5 core histone modification signals with the 100 bins information of each gene from the basic gene annotation file by *bedtools intersect* operation [48]. Finally, the input of each gene is a 100×5 matrix, where the 100 corresponds to the 100 bins and 5 represents the 5 core histone modifications.

3.2 Deep neural network methods

In this decade, deep neural networks have been widely used in many areas because of the remarkable performance. CNNs are powerful in extracting hierarchical features and have made great break through in computer vision domain, such as image classification [8], [9], [49], [50], face recognition [51], [52] and object detection [53], [54], [55]. RNNs are able to learn long term dependencies within data and have shown excellent results [13], [56] in pattern recognition tasks with sequential data.

CNNs are capable of extracting high level features but may fail to learn the long term connectivity about the data while RNNs achieve to model sequential data. [57] In order to preserve the strengths of both the CNNs and RNNs, and to complement their shortcomings, the CNNs and RNNs are combined into a hybrid model which is often referred as the CRNN. The combination of CNNs and RNNs has recently achieved excellent success in many areas such as image representation [58], speech recognition [57], sound event detection [59] and gene expression prediction [17].

In this thesis, we propose a CRNN model and apply it for the gene expression

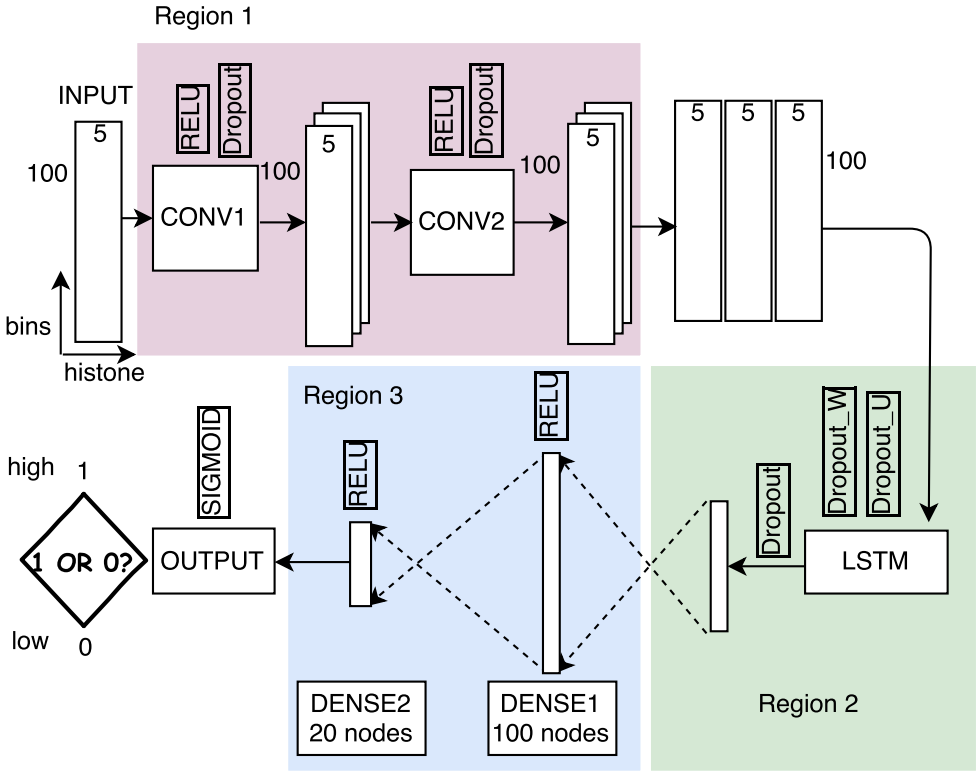

Figure 3.1 *Architecture of CRNN model(adapted from [17]).*

prediction tasks with histone modification signals. Additionally, two simplified versions: CNN and RNN, by removing the recurrent layer and convolutional layers from the CRNN respectively, and the state-of-the-art baseline: DeepChrome [7] are also discussed in this work.

3.2.1 Architecture of CRNN model

In this work, as each gene in our data is a 100×5 matrix, it can be regarded as an image. However, the 100 successive bins probably include dependency information. Thus, we are supposed to design a model that is capable of capturing spatial features while keeping the connectivity information.

We propose a CRNN method which is a hybrid model of CNN and RNN in this thesis. The architecture is presented in Figure 3.1. It consists of seven layers: one input layer, two convolutional layers, one LSTM layer, two fully connected layers and one output layer. After extracting features along both the bin axis and histone axis with the two convolutional layers, the features are mapped into 32 feature maps through ReLU activation function which has been proven to significantly accelerate the convergence of gradient decent [8]. These feature maps are stacked over the

histone axis before being fed into the LSTM layer. LSTM models long-term dependencies of data by using memory cells, and the drop regularization of inputs and recurrent state in the LSTM layer helps to reduce over-fitting. The recurrent layer is followed by two fully connected layers which are connected with ReLU activations. Finally, we get the binary prediction of each gene from the output layer with a sigmoid operator. Gene would be assigned as a high expression level if its prediction belonged to class "1", and vice versa. Moreover, in order to alleviate over-fitting, we adopt the dropout operation to each convolutional layer and LSTM layer of our CRNN model.

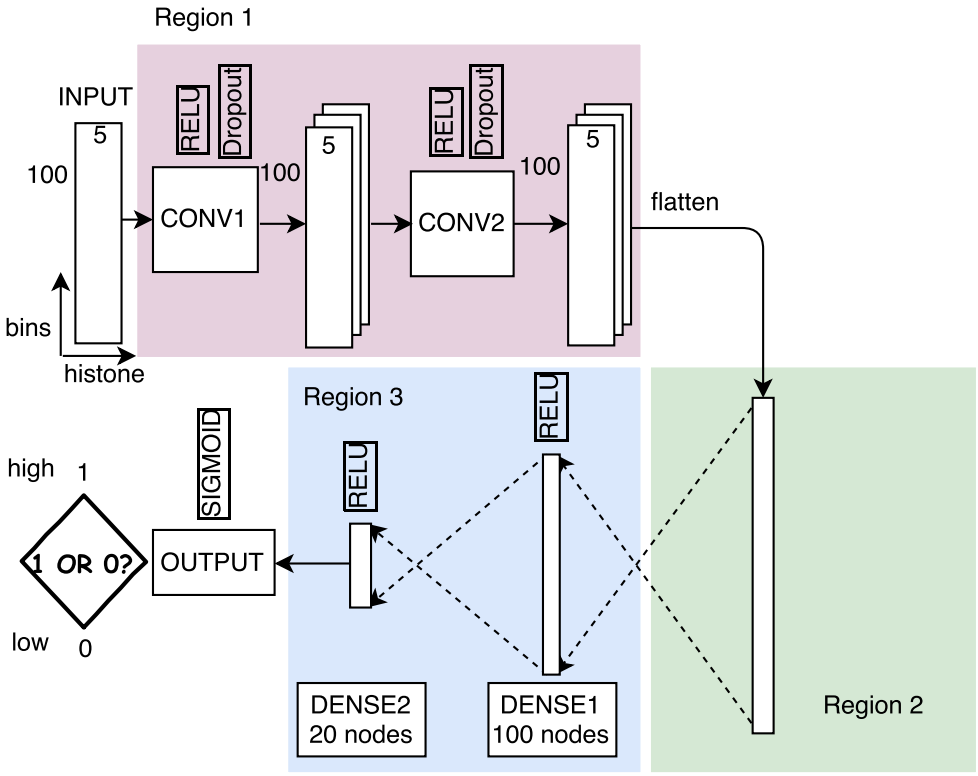

Figure 3.2 *Architecture of CNN model(adapted from [17]).*

3.2.2 Baseline models

In this subsection, the architecture of the three comparison models: CNN, RNN and DeepChrome are discussed.

CNN: One simplified variant of CRNN that we will study in this work: CNN, presented in Figure 3.2, is generated by removing the LSTM layer from the "Region 2" of Figure 3.1. It consists of one input layer, two convolutional layers, two fully connected layers and one output layer.

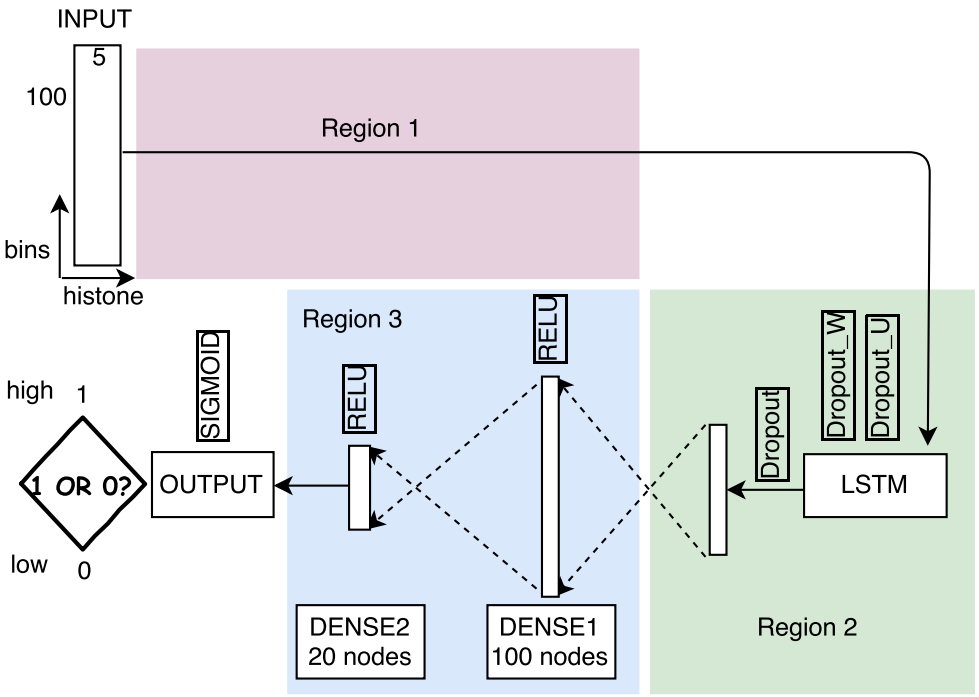

Figure 3.3 *Architecture of RNN model(adapted from [17]).*

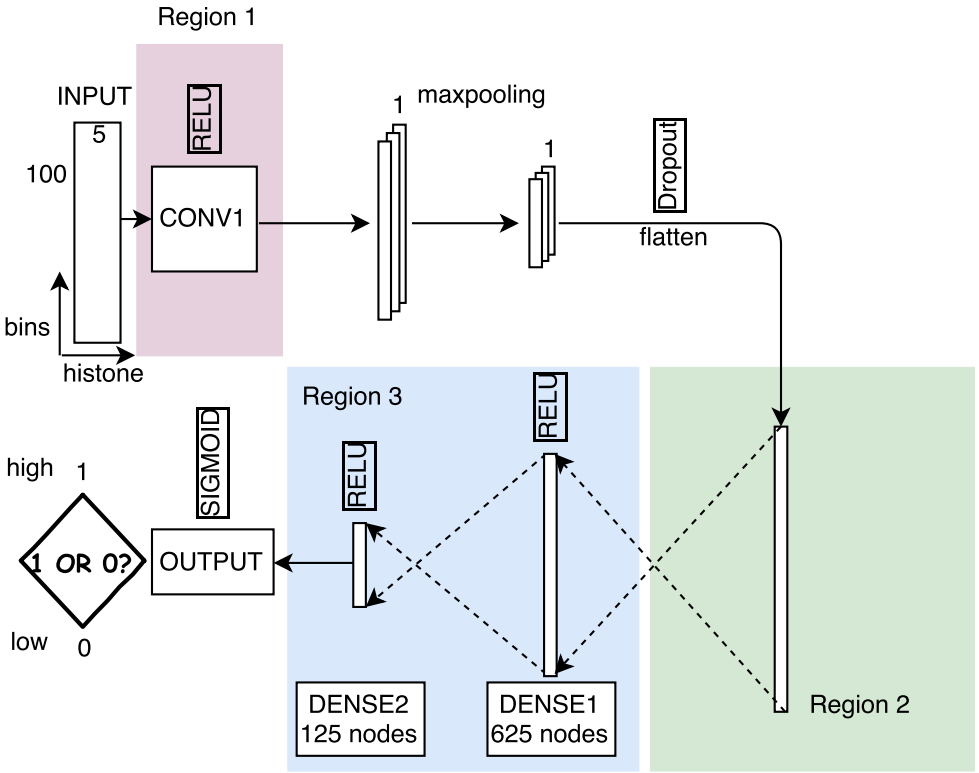

Figure 3.4 *Architecture of DeepChrome model used in this work(adapted from [17]).*

RNN: RNN is another simplified version of CRNN by removing the convolutional layers, denoted in "Region 1" of Figure 3.1. It is a model of one input layer followed with a LSTM layer, two dense layers and one output layer successively, which is illustrated in Figure 3.3.

DeepChrome: Singh introduced a convolutional network model named DeepChrome in [7], which is a model of one convolutional layer followed with a maxpooling operator, one dropout layer, two dense layers and one output layer. Additionally, the ReLU activation function is applied on the convolutional layer and dense layers for sparsity and reducing the likelihood of the gradient to vanish, and the sigmoid activation function is adopted in output layer for the final binary prediction in this work. Its architecture is shown in Figure 3.4.

4. RESULTS AND DISCUSSION

In this chapter, the different evaluation methods are introduced firstly. Then the experiments and their corresponding evaluation results are discussed. The analyses of the *Gene Expression Prediction* competition, which follows, are discussed in the end of this chapter.

4.1 Evaluation methods

4.1.1 Receiver Operating Characteristic

Receiver Operating Characteristic (ROC) curve is a fundamental and statistical tool for the evaluation of model performance. [60], [61] It is a commonly used way to quantify and visualize the performance of a binary classifier. There are four important concepts from a binary classifier when the two classes are labeled as positive "+1" and negative "-1":

- True Positive (TP): positive value is predicted to be positive.
- False Positive (FP): negative value is predicted to be positive.
- True Negative (TN): negative value is predicted to be negative.
- False Negative (FN): positive value is predicted to be negative.

A confusion matrix of binary classifier can be made in Figure 4.1 based on the above four terms, and several error matrices such as True Positive Rate (TPR) and False Positive Rate (FPR) are derived from the confusion matrix,

$$TruePositiveRate(TPR) = \frac{TP}{TP + FN} = \frac{TP}{P} \tag{4.1}$$

$$FalsePositiveRate(FPR) = \frac{FP}{FP + TN} = \frac{FP}{N} \tag{4.2}$$

where the P and N are the total positives and total negatives, respectively. The

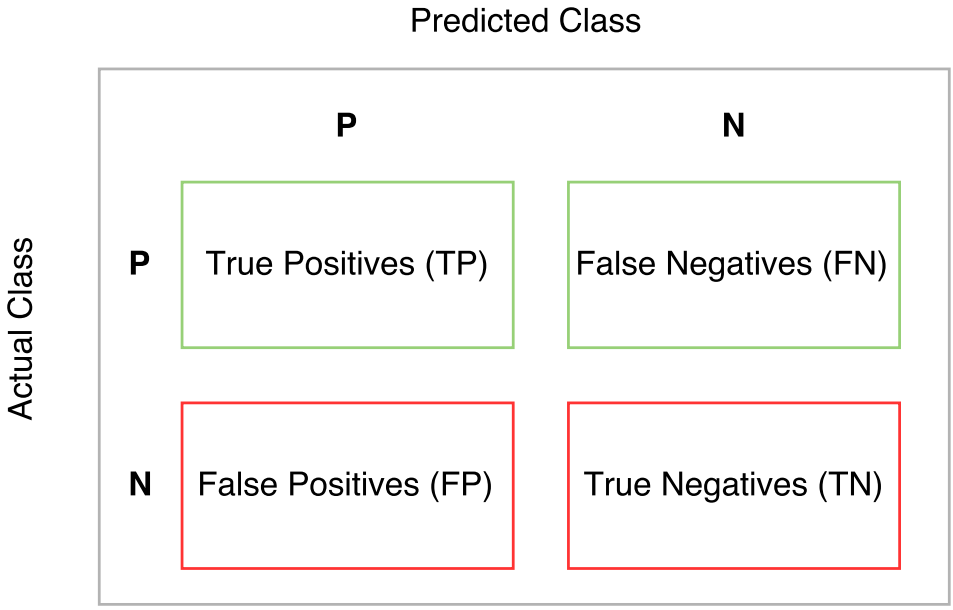

Figure 4.1 *Confusion matrix.*

TPR tells us how often does it predict "positive" when it is actually "positive", and also known as "sensitivity" or "recall". The FPR shows how often does it predict "positive" when it is actually "negative", and it can be calculated as 1 - "specificity", where the "specificity" indicates how often does it predict "negative" when it is actually "negative". [62]

ROC curve is generated by graphically plotting the TPR against the FPR while varying possible thresholds for assigning observations to the given classes. Figure 4.2 shows an example distribution of predicted probabilities of each class. The green and blue curves visualize the difference between the predicted probabilities and true statues. If we assume the height at probability: 0.3 is 40, at 0.5 is 10, it means that the probability of predicting the 40 samples to be positive are 0.3, and the true status for all the 40 items is negative. There are about 10 samples for which we predict a probability of 0.5, and half of them are positive and the other half are negative. Based on this example, we think that this classifier works good as it separates the classes well. Its ROC curve over all possible thresholds is drawn in Figure 4.3. In a ROC, the TPR is graphically plotted in function of the FPR for various decision thresholds. For example, if we set the threshold at 0.0 where is the black bar located in the distribution of predicted probabilities, the classifier classifies all items as positive (TPR = 1.0, FPR = 1.0) which corresponds to the red point at the top-right corner of the ROC curve in Figure 4.3. If dragging the threshold right, the corresponding TPR-FPR point will move to the left and down side. When

the threshold moves to 0.5 which is the default setting of most classifiers, all of the items above 0.5 are predicted as positive and the ones below 0.5 are negative. Its TPR-FPR point moves to the top-left corner (green point) of the ROC curve.

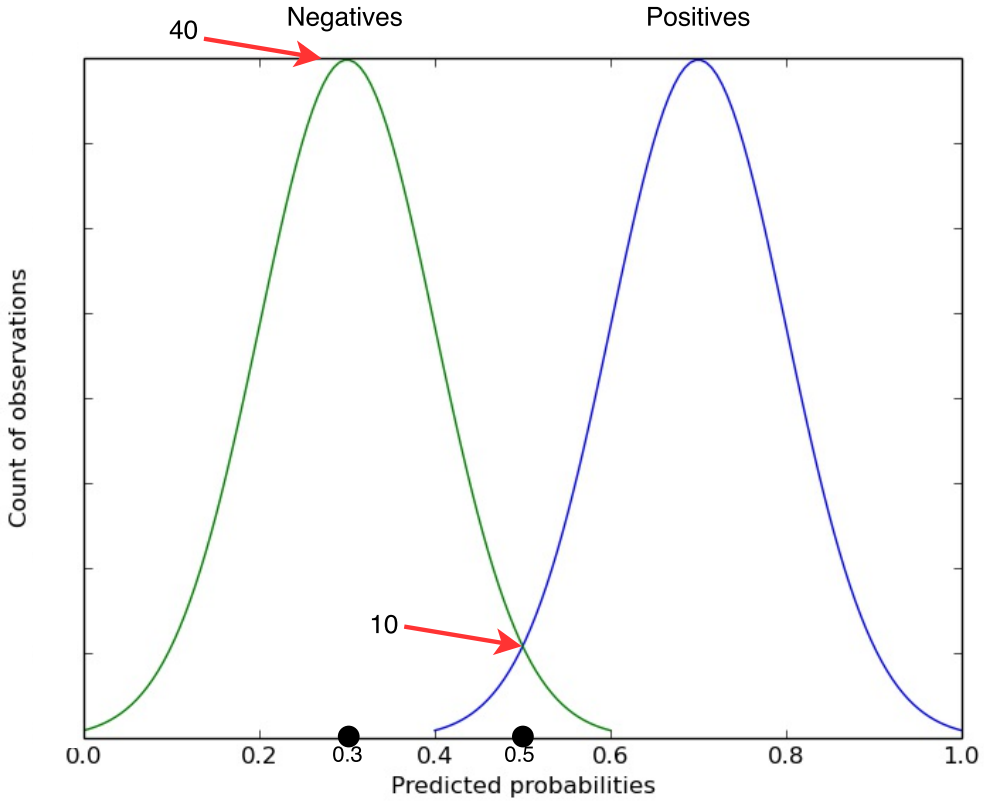

Figure 4.2 *Distribution of predicted probabilities.*

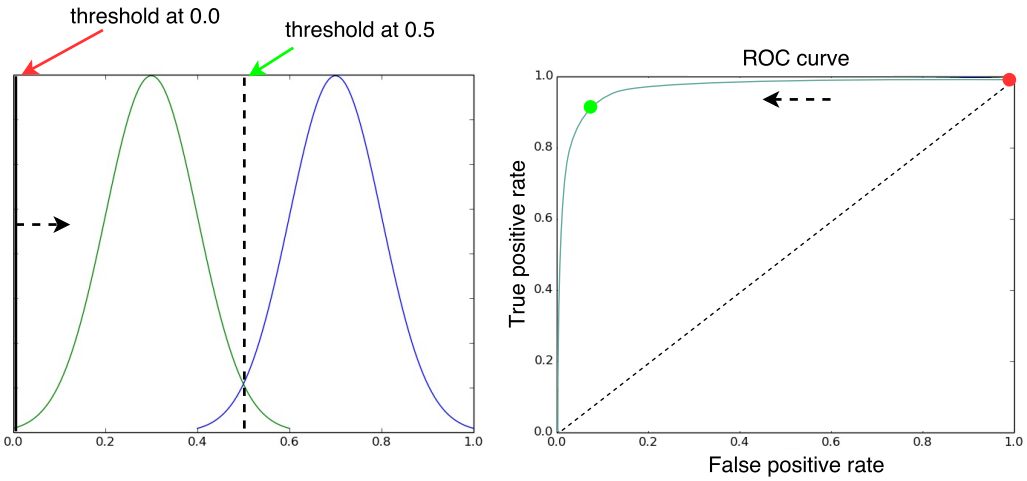

Figure 4.3 *ROC curve for the first distribution example.*

The given example in Figure 4.2 is a classifier that works good for classifying the two classes well. However, finding an optimal classifier usually takes time in reality.

The classification accuracy will become lower when the overlap between the two distributions becomes larger by moving the green distribution right, which is shown in Figure 4.4. This happens regardless of where we set the threshold. Its ROC curve can be generated by plotting the TPR versus FPR by varying all possible thresholds which range from 0 to 1.0. We can get even worse performance by moving the green distribution righter which is presented in Figure 4.5. This classifier does a pretty poor classification with a ROC curve that is close to the diagonal line which denotes random guessing. It infers that the performance of this classifier is no better than the random case.

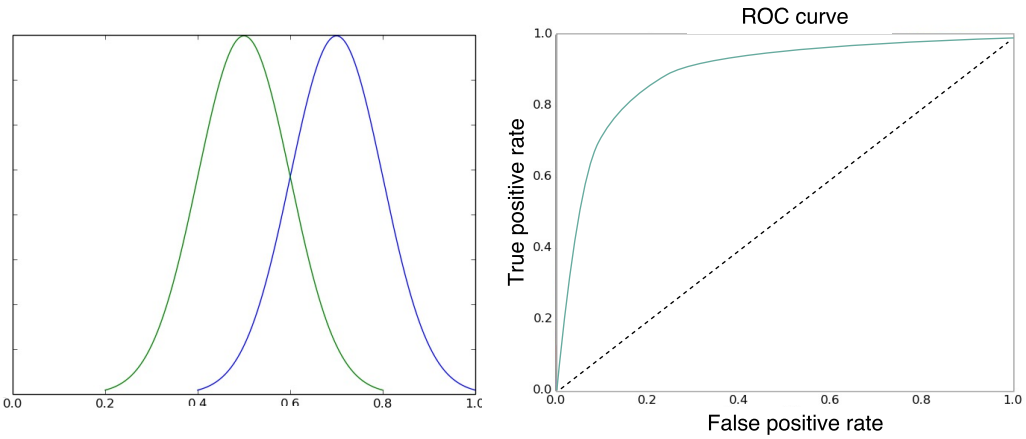

Figure 4.4 *ROC curve for the second distribution example.*

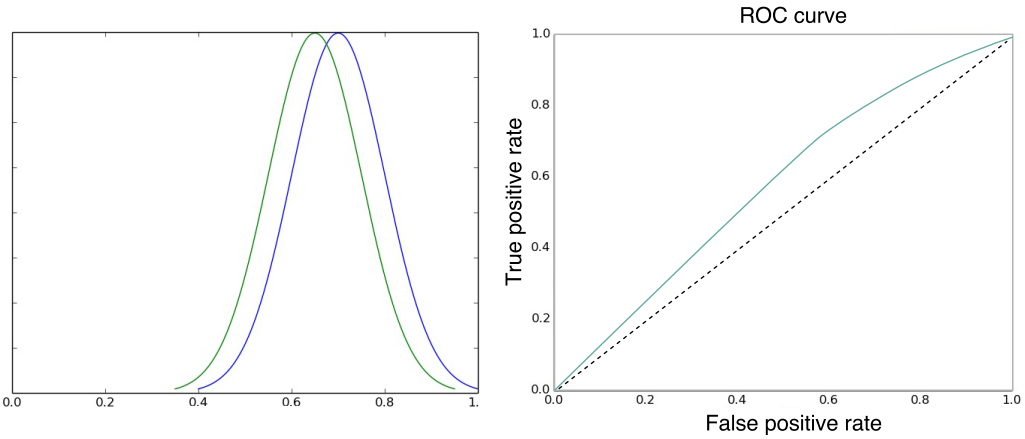

Figure 4.5 *ROC curve for the third distribution example.*

In Figure 4.6, the red ROC curve that hug the up left corner of the plot is the optimal one. The classifiers perform worse if their ROC curves are more closer to the black diagonal dashed line which represents the random guessing. The ROC curves are commonly used for visualizing and comparing the performance of classifier

methods. Moreover, Area Under the Curve (AUC) is a good way of quantifying the performance of a classifier from ROC. AUC is literally the percentage of the area under the curve. Its value ranges from 0 to 1.0. The larger the AUC scores are, the better the classifiers perform. Furthermore, the AUC is a very useful method when the dataset is highly unbalanced and when the predicted probabilities have a large variance or a "smooth" distribution. [61], [63]

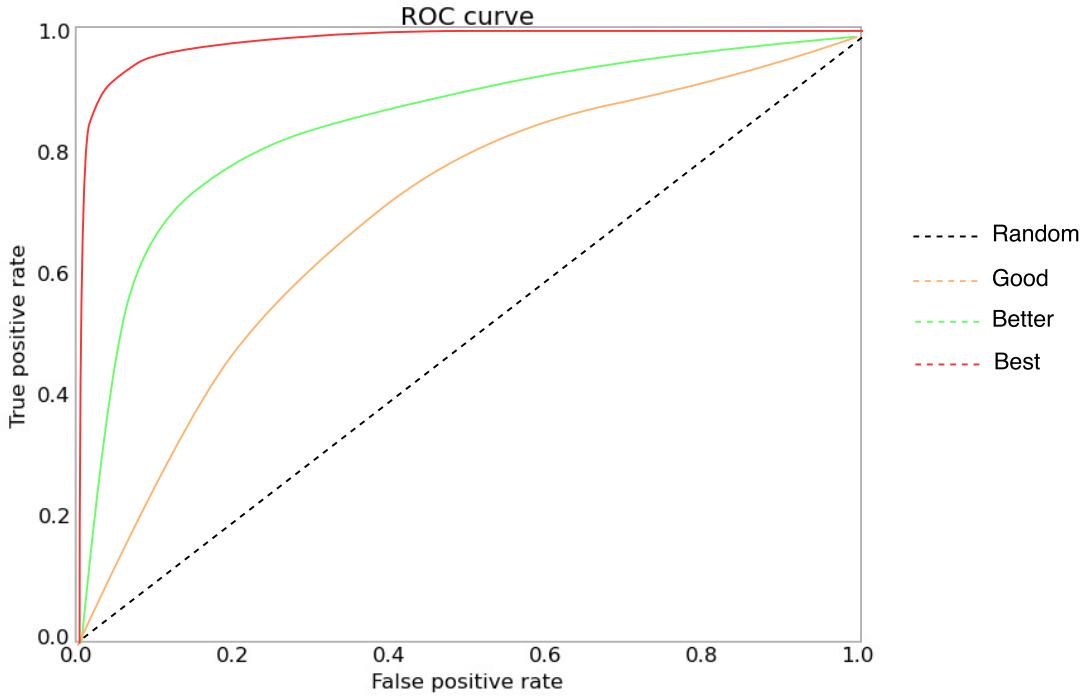

Figure 4.6 *Comparison of ROC curves.*

4.1.2 P-value

P-value is used to determine statistical significance in a hypothesis test. Null hypothesis is an essential concept for the interpretation of p-values. For each experiment, there is the difference between groups that the researchers are testing. In this work, the groups could be the effectiveness of the different studied models. It is always possible that the lack of difference happens between the groups, which is called the null hypothesis. A p-value is the probability of occurrence of an observed result when we assume the null hypothesis is true. Graphically, the p-value is the shaded magenta area in the tail of probability distribution, where the portion under the curve that past the observed point. It is illustrated in Figure 4.7.

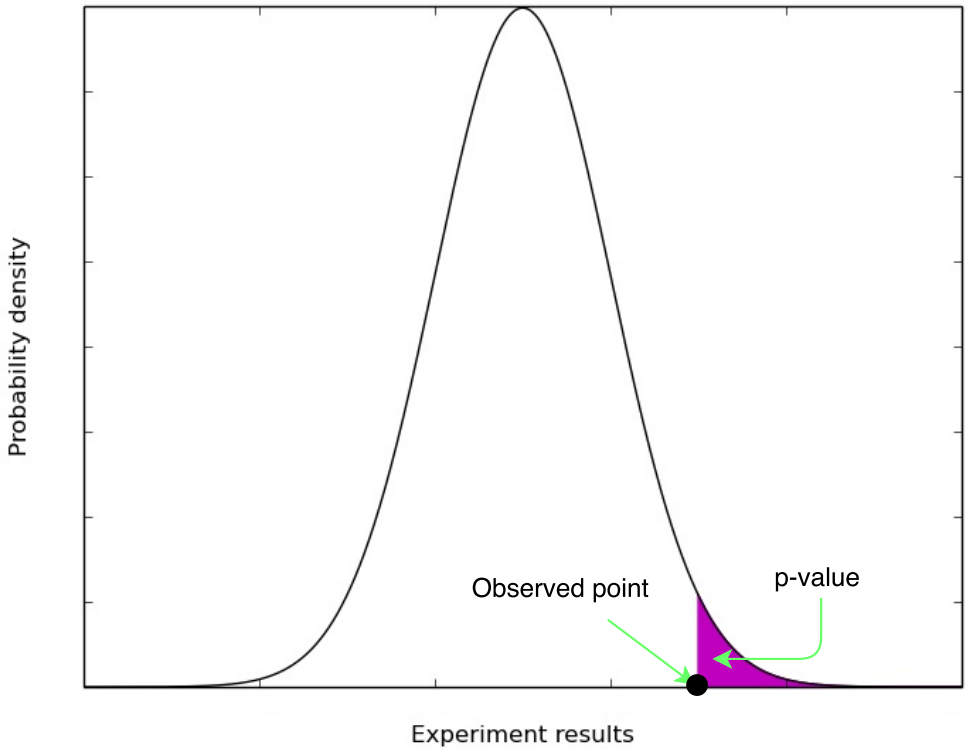

Figure 4.7 *P-value.*

P-value manages how compatible our data is with the null hypothesis. The alternative hypothesis is the opposite of the null hypothesis. In other words, alternative hypothesis is the one we believe when the null hypothesis is predicted to be untrue. A low p-value indicates that our data instances provide enough evidence that there is a significant difference between the compared groups. Hypothesis tests adopt a p-value to weigh the strength of the evidence, and are expressed as decimals to demonstrate the probabilities that the results could be random. There is a term named significance level (α) which is used to refer to a predefined probability while the term p-value is used to indicate a probability that we calculate from the experiment work. Conventionally, the significance levels of 5%, 1% and 0.1% are commonly set by researchers before examining the data instead of deriving from observational data or underlying hypothesis. A value of 5% is chosen for α in our work empirically. The p-value is range from 0 to 1.0 and interpreted as follows:

A low p-value ($\leq \alpha$) manifests strong evidence against the null hypothesis so that we could reject the null hypothesis.

A high p-value ($\geq \alpha$) manifests no significant result against the null hypothesis so that we fail to reject the null hypothesis.

A p-value close to significance level (α) is considered to be marginal.

Therefore, the smaller the p-value is, the more important our result is. Because it indicates that the null hypothesis under consideration probably fails to explain the observation adequately.

4.2 Experiments

The data that derived in Section 3.1 includes 19356 genes of 56 cell types. Thus, our task in this work is to develop a general model that can accurately predict gene expression levels from histone modification signals over 56 cell types. For each cell type, there are 19356 genes, and we use this data for training, validation and testing. Specifically, we partition 60% of all genes of each cell type from the dataset into training set, 20% of all genes into validation set and 20% into test set. Furthermore, we implement the studied models by Keras[1] with Theano[2].

We evaluate the proposed CRNN model with the histone modification signals and gene expression data from 56 cell types. Its performance is compared to two simplified variants: CNN and RNN, and baseline: DeepChrome which are discussed in Subsection 3.2.2. In the CRNN model, each convolutional layer uses 3×3 filter kernels and a stride of 3. The number of filters we choose for each convolutional layer is 32. Two fully connected layers with 100 and 20 nodes follow a LSTM layer which includes 32 units. The dropout strategy with 50% probability is applied to avoid network over-fitting. In the CNN and RNN models, they have almost the same setting with the CRNN model except the dropout rate of 20% for the RNN model. However, for the DeepChrome model, it has 50 5×10 kernels for convolution, and we adopt the stride of 5 in our work. The convolutional layer is followed by a maxpooling layer with a 5×5 window, and the number of hidden units in the two fully connected layers are 625 and 125.

As predicting the gene expression levels from histone modification signals in this work is a binary classification task, the studied models are finally tuned by minimizing the loss function L:

$$L = -\boldsymbol{y} \log(\hat{\boldsymbol{y}}) - (1 - \boldsymbol{y}) \log(1 - \hat{\boldsymbol{y}}) \tag{4.3}$$

which is specified as "binary_crossentropy". The loss L calculates the binary crossentropy between predictions $\hat{\boldsymbol{y}}$ and targets $\boldsymbol{y}$. The learning rate of 1e-3 is chosen for

[1] `https://keras.io/`
[2] `http://deeplearning.net/software/theano/`

the learning convergence. We employ the optimizer of Adam to the CRNN and RNN models, SGD to CNN and DeepChrome models. The batch size: 200 and epochs: 500 are set for training CRNN, CNN and RNN networks. Based on the experiments we implemented, the parameters of batch size : 1 and epochs: 100, similar parameters setting in [7], worked best for DeepChrome.

4.3 Results

The AUC scores over the 56 cell types of each model are visualized in Figure 4.8, and have been sorted based on an decreasing order of performance of the CRNN model. As we can clearly see, the blue curve on the top, which indicates the performance of the CRNN model, is above all the three other models for most cell types, and the CRNN, CNN and RNN models outperform the DeepChrome. Interestingly, the AUC scores of the four models follow a similar trend in predicting gene expression level over 56 cell types. In other words, some cell types are always predicted better or worse than others. For instance, cell types such as "E117", "E123", "E118" and "E055" always have a good prediction from all models. However, for the cell types like "E065", "E112", "E071" and "E094", all the models perform very poorly. This result gives us the insight that the signals of some cell types are very weak. Different models learned "weak" features from the weak signals and therefore perform poorly on them. This most likely reflects the biological complexity of the different cell types. Gene expression is probably regulated also by other factors except the five histone modifications included in this study, and therefore the regulation degree captured by the data that we derived from REMC database varies across cell types.

Additionally, the results of the studied models on test data are summarized in Table 4.1. It includes the average, maximum and minimum AUC scores over 56 cell types of each studied model, and their corresponding cell types of the maximum and minimum AUC score are also presented. The maximum value of each column is given in a bold format. The two rightmost columns are the p-values between each proposed model and DeepChrome model on the best "E117" and worst case "E065". The DeepChrome in this work means the experiment of applying "DeepChrome" model on the dataset that we derived from REMC database, and the information in the last row is the AUC scores of "DeepChrome" model from [7] which is named DeepChrome[1] here. As can be seen, the CRNN model has the largest average AUC score and the cell types that each model performs best ("E117", "E123", "E118") or worse ("E065") are almost always the same ones. This result also matches the information of similar models predicting trend that we observed from Figure 4.8.

Furthermore, for the best case "E117" and worst case "E065", the p-values between

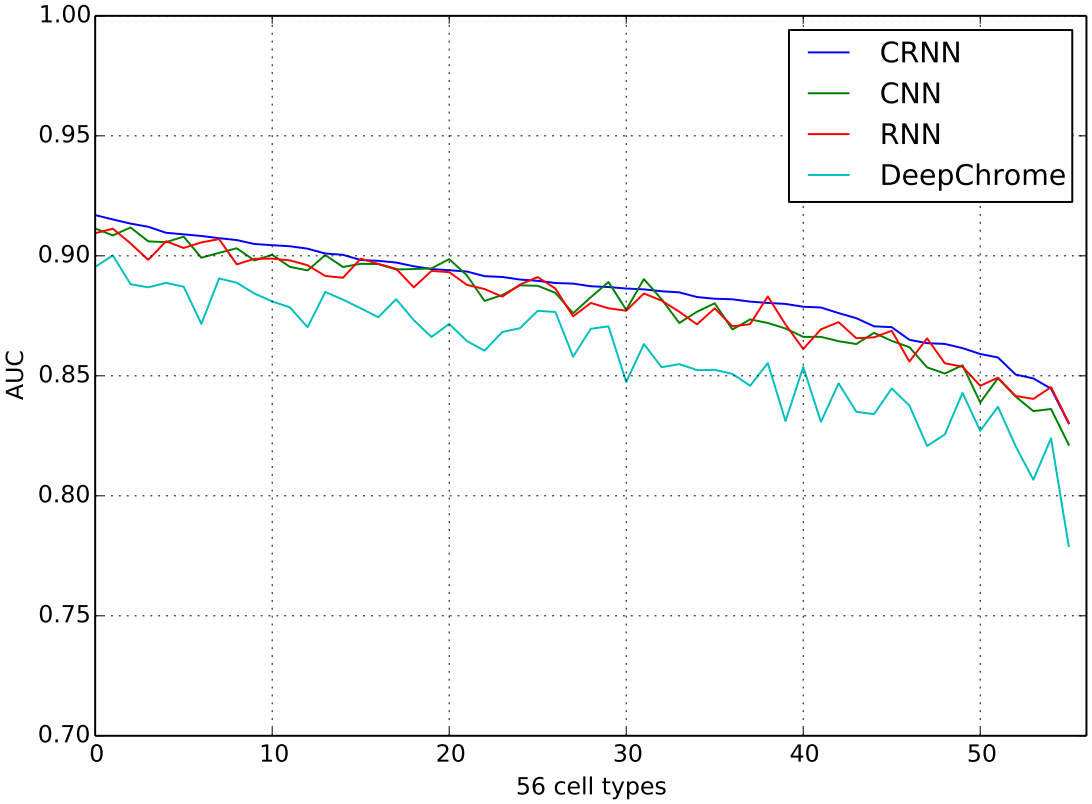

Figure 4.8 *AUC scores over 56 cell types.*

Models	AUC(Avg)	AUC(Max)/ cell type	AUC(Min)/ cell type	P-value (E117)	P-value (E065)
CRNN	**0.8861**	**0.9170**/E117	0.8302/E065	$< 10^{-5}$	$< 10^{-5}$
CNN	0.8799	0.9118/E118	0.8213/E065	0.0003	$< 10^{-5}$
RNN	0.8802	0.9113/E123	**0.8304**/E065	0.0001	$< 10^{-5}$
DeepChrome	0.8591	0.9002/E123	0.7790/E065	-	-
DeepChrome[1]	0.77	**0.94**/E123	0.66/E112	-	-

Table 4.1 *Performance of studied models on test dataset. (adapted from [17])*

each model and DeepChrome are also listed in Table 4.1. It is apparent that the p-values are smaller than the statistical significant level: 5% we chose in this work. The low p-values provide us enough evidence that the proposed CRNN, RNN and CNN models perform significantly better than DeepChrome, and this is not due to chance. Moreover, based on the AUC scores in Figure 4.8 and the ROC curves of the studied models on the worst case "E065" in Figure 4.9, we can clearly see that the three CRNN, CNN and RNN models achieve higher AUC scores and their ROC curves are more closer to the top left corner of the plot than the DeepChrome model. Thus, we would conclude that the presented CRNN, CNN and RNN models in our work outperform the DeepChrome model.

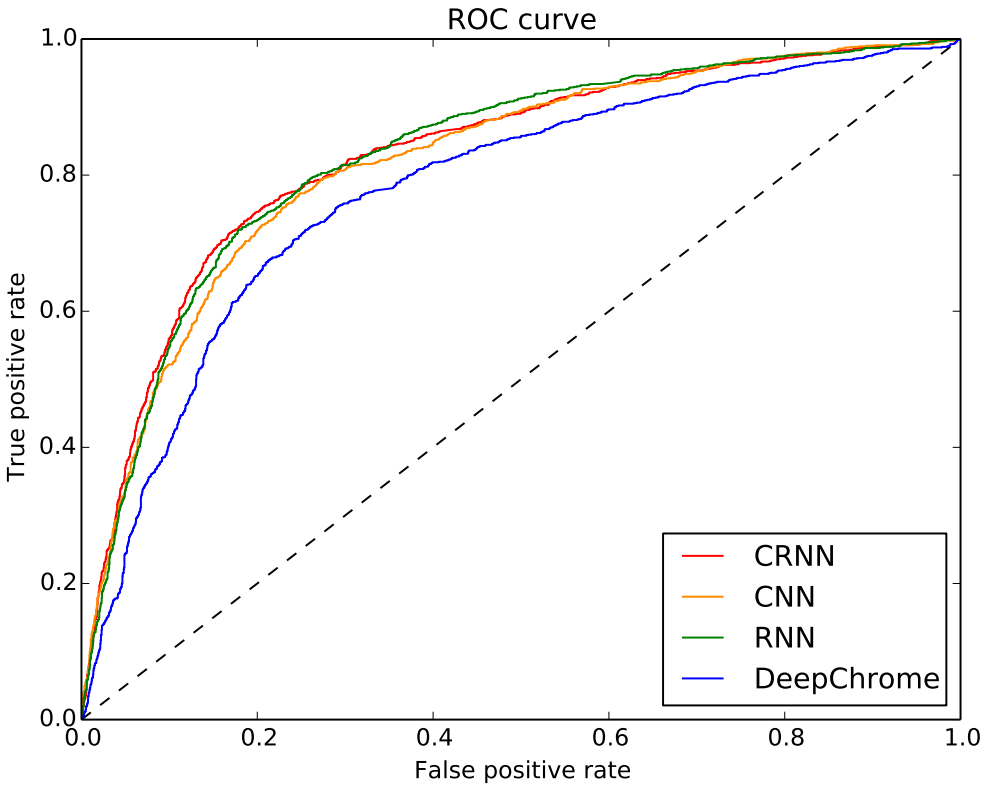

Figure 4.9 *The ROC curves of studied models of cell type "E065".*

Our performance clearly exceeds the result of DeepChrome[1], which has an average AUC score of 0.77 [7]. For this large improvement, we have two possible assumptions. On the one hand, as the approach that we used in Section 3.1 for generating the data is the feature intersection from BEDTools, our data captures the more important information that better represents the effect of the five histone modifications signals. Thus, though applying the DeepChrome model with the similar setting in [7] to our data, we still can get a good performance with average AUC score of 0.8591 as shown in Table 4.1. One the other hand, the different model architectures enable this large improvement also. For instance, though both of the DeepChrome and the CNN model that we discussed in Section 3.2 include the convolutional networks, their model architectures are totally different. Therefore, different models learned features with different hierarchical levels, and variant model settings enable the models fit into the data to different extent. This allows the proposed CNN model to perform better than DeepChrome on the same dataset.

4.4 Overview of gene expression prediction competition

Machine learning gains its popularity recently with its state-of-the-art performance in providing promising solutions for a wide range practical problems of data science. *Kaggle* is an online platform for organizing competitions with statistic analysis and predictive-modeling. An organization is capable of arranging with *kaggle* to propose a problem with their datasets, and the participants from various backgrounds could post their proposed solutions on *kaggle* which usually brings innovation to the organizations. Organizers usually arrange the competitions with different purposes and offer some pre-agreed "prize" as reward for the winning solutions. *Kaggle in Class* is a free of charge service example provided by *kaggle* for academic institutions with the purpose of education development.

The competition of *gene expression prediction* that we will discuss in this thesis was organized on *Kaggle in Class*. In consideration of various hardware configuration and computing ability of different students, just one cell type was finally chosen in this competition for simplicity. Instead of coming up with a general solution that fit well for all 56 cell types, the goal of this competition is to develop a model for accurate gene expression prediction on cell type "E047". Accordingly, just the corresponding histone modification signals and RNA-sequence (labels) data of cell type "E047" was used in this competition. The training dataset of this competition includes 15485 genes while the test set consists of 3871 genes. Each gene is a matrix of 100 bins and 5 histone modification signals. The corresponding labels of training set were provided for the participants while the labels of test data were kept unseen for scoring purpose of the submissions to competition.

The competition participants were expected to predict the likelihood (score between 0 and 1.0) of having a high expression level (label = 1) for each gene. It is also allowed to submit predicted category (high gene expression level: 1 or low gene expression level: 0), but the score will be less than with well justified likelihoods represented as real numbers. As the evaluation metric in this competition is the weighted AUC which assesses the accuracy based on the area under the ROC curve. It ranks the submissions based on the order of predicted likelihoods and lists the result on leaderboard. *Kaggle* involves a "supervising" mechanism for checking whether the participants overfit their models with the 50% public test data or not. It has the public leaderboard and private leaderboard. The public leaderboard is calculated on approximately 50% of the test data, and the final results will be listed on private leaderboard based on the other 50% left. Before the end date of this competition, only the public leaderboard is available to the participants and the private leaderboard would be published after the deadline of competition. Thus, a

high score on the public leaderboard is still possibly reaching a low score on the private leaderboard.

4.4.1 Competition submissions

This *gene expression prediction* competition[3] was organized on *Kaggle in Class* from 9 January 2017 to 5 March 2017. All the students who signed up the pattern recognition course from Tampere University of Technology were warmly encouraged to take part in this competition. Moreover, this competition was initialized as public so that any registered user can download and submit entries. During the period of two months, we received 888 entries from 105 teams with 184 players. Scikit-learn[4], XGBoost, Keras and Tensorflow[5] are the most commonly used libraries and frameworks in this competition. The weighted AUC scores and submitted entries numbers from the top 5 teams are listed in Table 4.2. As we can see, it consists of the AUC scores on both the private and public leaderboards for the top 5 teams and one benchmark submission. The results of the proposed CRNN model (a general CRNN model we studied for 56 cell types) on the cell type "E047" are also included. The one who took the first place is marked as bold: 0.94799 and 0.94160 on private and pubic leaderboard, respectively. Besides, there are very subtle score differences among these AUC numbers especially the four teams in the middle rows of Table 4.2. Additionally, each team solution outperforms the benchmark that the competition provided, and they endeavored to try to improve their final scores and this can be reflected from the submitted entries numbers.

Team Name	AUC score (private)	AUC score (public)	Entries
AndrewChang	**0.94799**	**0.94160**	5
Group32	0.92932	0.91435	37
Group36	0.92802	0.90899	13
Group40	0.92787	0.91557	36
Group28	0.92778	0.91529	43
CRNN	0.92515	0.91031	-
Benchmark	0.86540	0.87467	-

Table 4.2 *The weighted AUC scores from the top 5 teams, proposed CRNN model and one benchmark.*

Specifically, the ideas that the top 5 teams used are discussed as follows:

[3] `https://inclass.kaggle.com/c/gene-expression-prediction`
[4] `http://scikit-learn.org/stable/`
[5] `https://www.tensorflow.org/`

Andrew Chang: Andrew Chang firstly reshaped each gene of training set into a wide raw with 500 features (from a 5 × 100 matrix to a 1 × 500 vector). Then 5 additional features which are the sum of different histone modification marks over all bins were added as the total effect of all histone marks. Finally, each gene is a 1 × 505 vector. Besides, XGBoost was applied to train the model with training dataset and GridSearchCV[6] was adopted to fine tune the hyper-parameters. The 505 features were used to train the first layer of model, and the additional 5 features were proven to be the most important features. Then the corresponding predicted label was appended to the 505 features, and now each gene would be a 1 × 506 vector which can be used to train the second layer of model. This model achieved the highest scores of 0.94799 and 0.94160 on the private and public leaderboard, respectively.

Group 32: The submission of Group 32 was obtained by training a model with similar architecture of DeepChrome network through 10 stratified folds on the provided training data and taking the median of the predictions. The data was preprocessed by normalization before feeding into the model. In their model, one dimension convolution operation with filter length: 11 was applied. Dropout rate with 75% and 15% probabilities were adopted to avoid over-fitting before and after the first dense layer with 590 nodes. Two more dense layers with 60 and 78 nodes kept themselves as the successors, and followed by a dropout regulation of 30%. Their model obtained the scores of 0.92932 that sitting on private leaderboard and 0.91435 on the public leaderboard.

Group 36: Group 36 achieved a good score of 0.92802 on private leaderboard using a network that consisted of two convolutional layers with maxpooling operation and dropout rate of 50%, and 2 dense layers. The first convolutional layer used a 1 × 1 convolution to create 50 feature maps. One maxpooling layer with the window size of 1 × 2 was combined with this first convolutional layer. The second convolutional layer used a 5 × 5 convolution to again create 50 feature maps that will passed through a maxpooling layer of size 1 × 5. The output of this layer was then flattened and passed into a dense layer of 625 hidden nodes that was followed by a dense layer of 125 nodes. Finally, a dense layer of 2 nodes was connected at the end of the network in order to do the final binary classification. Furthermore, they used the ReLU as activation for all of the layers except the last one, which used SoftMax [33].

Group 40: A prediction accuracy of 0.92787 was achieved by stacking models of logis-

[6] `http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html`

tic regression, K-nearest neighbor, CNNs, RNNs and ensemble methods (ExtraTrees, Random Forest, XGBoost), and yielded the 4th place on the private leaderboard. Group 40 also generated additional features which were composed of the mean, standard deviation, percentage of zeros per sample and mean, standard deviation per marker per sample.

Group 28: Group 28 achieved the scores of 0.92778 and 0.91529 by picking up the mean of the AUC scores from the developed two deep learning models. One is a convolutional network with 6 convolutional layers, connected with a maxpooling layer every three convolutional layers, and two fully connected layers. Another one is a GRU based model which includes two GRU layers with dropout regulation and one dense layer of 180 nodes.

4.4.2 Analysis of competition submissions

Based on the submissions (entries) and final scores of each group, we would say that each group has tried different machine learning and statistical methods that they learned from that pattern recognition course from Tampere University of Technology, and also brought their innovation to this competition. As is shown from Table 4.2, Andrew Chang achieves the best scores and each group performs better than the benchmark which is a model based on KNN. In order to compare this visually and more concretely, their ROC curves and p-values on entire test dataset are illustrated in Figure 4.10 and Table 4.3, respectively. Figure 4.10 shows that the ROC curve of the model from Andrew Chang, which marked as red, is close to the top left corner most and apparently better than the others. Apparently, each curve outperforms the benchmark. However, it is hard to tell which one of Group 32, Group 40, Group 28, Group 36 and CRNN models works better than the others left just from the ROC curves in this figure.

From the upper triangle of Table 4.3, we can observe that the AUC differences between every two models from Group 32, Group 40, Group 28, Group 36 and CRNN model are very small, especially the ones between Group 32 and Group 40, Group 32 and Group 28, Group 40 and Group 28, Group 36 and CRNN model, which are marked as blue. If we set the significance level (α) to be 5%, the p-values between each model are listed in the lower triangle of Table 4.3. The p-values, which are larger than α, are marked as red while the p-values that smaller than 5% are shown in green color. Based on the p-values which are smaller than α, Table 4.3 indicates that the additional features and the XGBoost method of Andrew Chang outperform the others. The approaches of Group 32, Group 40 and Group 28 perform significantly better than Group 36, studied CRNN model and the benchmark. Group 36 and

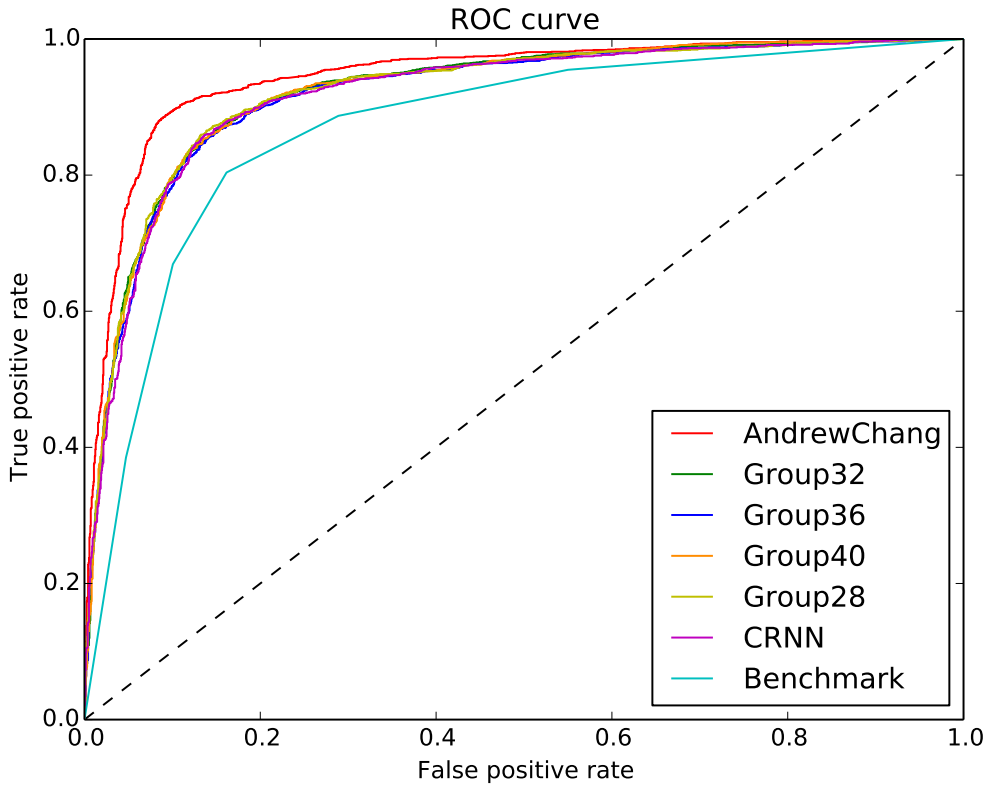

Figure 4.10 *The ROC curves of cell type "E047" from top 5 submissions, CRNN model and KNN benchmark.*

CRNN model exceed the benchmark also. By contrast, the p-values among each model of Group 32, Group 40 and Group 28 are larger than α, which claims that there is no significant difference between every two of them. Similarly, the null hypothesis is also "true" in the case of Group 36 and the studied CRNN model as the p-value between them is larger than %5. Moreover, these "red" p-values in the lower triangle are symmetric and match well with the small "blue" AUC differences in the upper triangle. This indicates the lack of differences of the compared models.

Team Name	AndrewChang	Group32	Group40	Group28	Group36	CRNN	Benchmark
AndrewChang	-	0.0229	0.0232	0.0232	0.0263	0.0271	0.0749
Group32	0.0000	-	0.0003	0.0004	0.0034	0.0042	0.0520
Group40	0.0000	0.8013	-	0.0001	0.0031	0.0039	0.0517
Group28	0.0000	0.7733	0.9564	-	0.0030	0.0038	0.0516
Group36	0.0000	0.0063	0.0213	0.0277	-	0.0008	0.0486
CRNN	0.0000	0.0022	0.0056	0.0052	0.5836	-	0.0478
Benchmark	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	-

Table 4.3 *P-values and AUC differences of cell type "E047" from top 5 submissions, CRNN model and KNN benchmark.*

4.4.3 Communication with customers

Weekly group meeting with the customers from University of Tampere was organized as an extension of this competition. During each meeting, team members and customers discussed about the work that the team has done since previous meeting and future plan before the next meeting. Besides, as most of the students have no knowledge background of biology, the customers also introduced and explained the data and tasks more concretely to each group.

The students gave the feedback that attending the meetings with customers helped them understand the project better, especially the data part that related to biology. For example, the students learned how the 100×5 matrix of each gene was generated from original biological files, and the significance of the TSS to feature selection. The conversations enlightened the students with insights of data analysis and model constructions. Moreover, the weekly meeting also motivated each group member to work and discuss more during this competition.

4.4.4 Learning assessment

Robert M. Gagn proposed a learning hierarchy system which helps to classify different learning types, in terms of the degree of complexity of the involved mental processes. Eight basic learning types are identified and arranged in Figure 4.11. The complexity increases with the increasing orders of learning. The lowest four learnings such as signal learning, stimulus-response learning, Chaining and the verbal association are more focus on the behavioral aspects of learning while the higher four learnings: discrimination learning, concept learning, rule learning and problem solving concentrate on the cognitive aspects more. Additionally, Gagn also indicated that the higher level of learning is built and based on the lower levels in this hierarchy. [64]

Learning assessment is tied to learning goals and standards. The pattern recognition course from Tampere University of Technology tried to reach its expected learning outcomes by systematic design and management. To enable the students to understand the principles of selected statistical, pattern recognition and machine learning approaches in signal processing related problems, and to help them gain the ability of applying different methods to real problems using modern Python tools such as Scikit-Learn and Keras, this course was finally formed in a combination of lectures, video recording of each class, weekly exercises, competition and exam. The feedback from students was mainly collected from the weekly exercise, competition

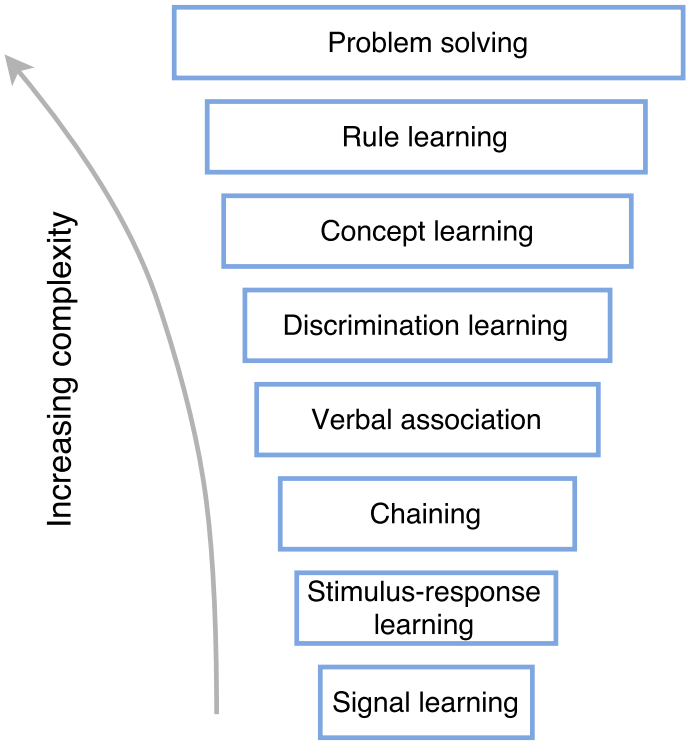

Figure 4.11 *Eight hierarchy of learning from Gagn.*

reports and final exam. By comparing and analyzing the feedback form students with the eight hierarchies of learning from Gagn, we would say that most of the learning levels are covered by this course. Furthermore, the learning assessment results show us what the students can learn from this course. For example, signal learning, stimulus-response learning, discrimination learning and concept learning are involved in the course lectures, video recording and weekly exercises. In the meanwhile, the chaining, verbal association, rule learning and problem solving are involved with weekly exercise and competition execution which is the extension of this course. Moreover, their feedback allows the course organizers to adjust the design and instruction of this course.

4.5 Discussion

As a result, the proposed CRNN, CNN and RNN models outperform the DeepChrome, and the CRNN had the best performance. The convolutional layers in CRNN help capture the features with higher hierarchies across bins and histone signals while the LSTM layer learns the connectivity along a longer range of bins, and this makes the CRNN stand out. Despite the better performance of CRNN in this work, the subtle average AUC scores differences between the CRNN model and CNN, RNN models in Table 4.1 suggest us more further research work on the biological complexity of different cell types in the future.

As a auxiliary extension part of the pattern recognition course, the competition *Gene Expression Prediction* was held successfully during this course. Students participated in this competition actively and proposed their remarkable solutions. The champion group adopted a method of scalable distributed gradient boosting system: XGBoost with additional features, which achieved good scores of 0.94799 and 0.94160 on the private and public leaderboards, respectively. Actually, this good performance is better than the AUC score of the developed CRNN model on the specific cell type "E047" in this thesis. As we know, only the data of one cell type "E047" was involved in this competition for the purpose of easy implementation. However, the studied CRNN is a general model that works well for 56 cell types. Further research of specific cell type with the CRNN model could be considered as the future work.

5. CONCLUSIONS

Different machine learning methods are becoming more and more popular in analyzing biological data and solving relative problems. They do this with excellent accuracy and high efficiency. Deep learning is especially popular, because of its remarkable performance. In this thesis, we compared the state-of-the-art baseline: DeepChrome model with the proposed CRNN model. Besides, the two simplified variants: CNN and RNN, by removing the recurrent layer and convolutional layers from the CRNN model respectively, were also discussed. As a result, the CRNN, CNN and RNN models outperform the DeepChrome model significantly. Additional, based on the Figure 4.8, we observed that the AUC scores curve of the CRNN model is above the curves of other models across most of the 56 cell types. We would say the CRNN model performs better than its simplified versions: CNN and RNN in general, despiting the small differences among their average AUC scores in Table 4.1. As we believe that the CRNN model could integrates the strengths of its both variants while complementing their shortcomings. Moreover, the biological complexity of different cell types may greatly affect the final average value of the AUC scores across 56 cell types. This issue is worth investigating more in the future work.

Based on experiments result, we conclude that both of the data preprocessing work and model architecture play an important role in our binary classification predicting task. For instance, the same DeepChrome model on different datasets achieved two scores with large difference, the model with our date obtained a big improvement which tells us the importance of data preprocessing. Interestingly, the studied four models: CRNN, CNN, RNN and DeepChrome follow a similar gene expression prediction trend. In other words, some cell types are always predicted well with high AUC scores while some cell types such as "E065" and "E112" are predicted worse with all the models. This reflects the analyzing complexity of biological data. As data analysis is one of the most important factors of machine learning applications, the study of data extraction and analysis is worth paying more attention in the future work on this topic. Though the DeepChrome model is one type of convolutional networks, it performs worse than the CNN model that we discussed before with same dataset. This is due to the architecture and parameter differences of the two models. Thus, we indicate that the modeling work affects the performance

significantly as well.

The *Gene Expression Prediction* competition, along with weekly exercises, video recording and lectures, was considered as the auxiliary portion of this pattern recognition course from Tampere University of Technology. This competition was successfully organized with the ebullient participation of students. The theoretical concepts of the selected statistical, pattern recognition and machine learning approaches that the students learned from the lectures were applied adequately into practice during this competition by solving the real problems of gene expression prediction with Python. As the course and this competition was jointly organized by Tampere University of Technology and University of Tampere, its success paved the way for the merger of three universities in Tampere to some extent.

BIBLIOGRAPHY

[1] Strahl BD, Allis CD. The language of covalent histone modifications. Nature. 2000 Jan 6;403(6765):41-5.

[2] Bannister AJ, Kouzarides T. Regulation of chromatin by histone modifications. Cell research. 2011 Mar 1;21(3):381-95.

[3] Dong X, Weng Z. The correlation between histone modifications and gene expression. Epigenomics. 2013;5(2):113-116.

[4] Karli R, Chung HR, Lasserre J, Vlahoviek K, Vingron M. Histone modification levels are predictive for gene expression. Proceedings of the National Academy of Sciences. 2010 Feb 16;107(7):2926-31.

[5] Cheng C, Yan KK, Yip KY, Rozowsky J, Alexander R, Shou C, Gerstein M. A statistical framework for modeling gene expression using chromatin features and application to modENCODE datasets. Genome biology. 2011 Feb 16;12(2):R15.

[6] Dong X, Greven MC, Kundaje A, Djebali S, Brown JB, Cheng C, Gingeras TR, Gerstein M, Guig R, Birney E, Weng Z. Modeling gene expression using chromatin features in various cellular contexts. Genome biology. 2012 Sep 5;13(9):R53.

[7] Singh R, Lanchantin J, Robins G, Qi Y. DeepChrome: deep-learning for predicting gene expression from histone modifications. Bioinformatics. 2016 Sep 1;32(17):i639-48.

[8] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. InAdvances in neural information processing systems 2012 (pp. 1097-1105).

[9] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. 2014 Sep 4.

[10] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015 (pp. 1-9).

[11] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2015 (pp. 3431-3440).

[12] Hinton G, Deng L, Yu D, Dahl GE, Mohamed AR, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath TN, Kingsbury B. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine. 2012 Nov;29(6):82-97.

[13] Graves A, Mohamed AR, Hinton G. Speech recognition with deep recurrent neural networks. InAcoustics, speech and signal processing (icassp), 2013 ieee international conference on 2013 May 26 (pp. 6645-6649). IEEE.

[14] Alipanahi B, Delong A, Weirauch MT, Frey BJ. Predicting the sequence specificities of DNA-and RNA-binding proteins by deep learning. Nature biotechnology. 2015 Aug 1;33(8):831-8.

[15] Kraus OZ, Ba LJ, Frey B. Classifying and segmenting microscopy images using convolutional multiple instance learning. arXiv preprint arXiv:1511.05286. 2015 Nov 17.

[16] Kundaje A, Meuleman W, Ernst J, Bilenky M, Yen A, Heravi-Moussavi A, Kheradpour P, Zhang Z, Wang J, Ziller MJ, Amin V. Integrative analysis of 111 reference human epigenomes. Nature. 2015 Feb 19;518(7539):317-30.

[17] Zhu L, Kesseli J, Nykter M, Huttunen H. Predicting Gene Expression Levels from Histone Modification Signals with Convolutional Recurrent Neural Networks. Joint Conference of European Medical and Biological Engineering Conference (EMBEC) and Nordic-Baltic Conference on Biomedical Engineering and Medical Physic (NBC). Accepted on 11, April, 2017.

[18] Mitchell T. Machine learning (mcgraw-hill international edit). 1997.

[19] Ayodele TO. Types of machine learning algorithms. INTECH Open Access Publisher; 2010 Feb 10.

[20] Michalski RS, Carbonell JG, Mitchell TM, editors. Machine learning: An artificial intelligence approach. Springer Science & Business Media; 2013 Apr 17.

[21] Kanungo T, Mount DM, Netanyahu NS, Piatko CD, Silverman R, Wu AY. An efficient k-means clustering algorithm: Analysis and implementation. IEEE transactions on pattern analysis and machine intelligence. 2002 Jul;24(7):881-92.

[22] Kohonen T. The self-organizing map. Neurocomputing. 1998 Nov 6;21(1):1-6.

[23] Vesanto J, Alhoniemi E. Clustering of the self-organizing map. IEEE Transactions on neural networks. 2000 May;11(3):586-600.

[24] Kotsiantis S, Kanellopoulos D. Association rules mining: A recent overview. GESTS International Transactions on Computer Science and Engineering. 2006 Jan;32(1):71-82.

[25] Chapelle O, Scholkopf B, Zien A. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. IEEE Transactions on Neural Networks. 2009 Mar;20(3):542-542.

[26] Zhu X, Goldberg AB. Introduction to semi-supervised learning. Synthesis lectures on artificial intelligence and machine learning. 2009 Jun 8;3(1):1-30.

[27] Deza MM, Deza E. Encyclopedia of distances. InEncyclopedia of Distances 2009 (pp. 1-583). Springer Berlin Heidelberg.

[28] Hsu CW, Chang CC, Lin CJ. A practical guide to support vector classification. 2003: 1-16.

[29] Breiman L. Random forests. Machine learning. 2001 Oct 1;45(1):5-32.

[30] Friedman JH. Greedy function approximation: a gradient boosting machine. Annals of statistics. 2001 Oct 1:1189-232.

[31] Chen T, Guestrin C. Xgboost: A scalable tree boosting system. InProceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 2016 Aug 13 (pp. 785-794). ACM.

[32] Pafka S. Simple/limited/incomplete benchmark for scalability, speed and accuracy of machine learning libraries for classification. Available at: https://github.com/szilard/benchm-ml 2016 Dec. [Accessed 26 April, 2017].

[33] Haykin S, Network N. A comprehensive foundation. Neural Networks. 2004 Feb;2(2004):41.

[34] Stanford, University. CS231n Convolutional Neural Networks for Visual Recognition. [image] Available at: http://cs231n.github.io/convolutional-networks/#pool 2016. [Accessed 28 Mar, 2017].

[35] Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Cognitive modeling. 1988 Oct;5(3):1.

[36] Kingma D, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014 Dec 22.

[37] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998 Nov;86(11):2278-324.

[38] Michael N. Deep learning. Available at: http:// neuralnetworksanddeeplearning.com/chap6.html 2017 Jan; Chapter 6. [Accessed 29 Mar, 2017].

[39] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015 May 28;521(7553):436-44.

[40] Karpathy A, (2015). The Unreasonable Effectiveness of Recurrent Neural Networks. [image] Available at: http://karpathy.github.io/2015/05/21/rnn-effectiveness/ [Accessed 29 Mar, 2017].

[41] Chung J, Gulcehre C, Cho K, Bengio Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555. 2014 Dec 11.

[42] Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation. 1997 Nov 15;9(8):1735-80.

[43] Chung J, Glehre C, Cho K, Bengio Y. Gated Feedback Recurrent Neural Networks. InICML 2015 Feb 9 (pp. 2067-2075).

[44] Wang Z, Zang C, Rosenfeld JA, Schones DE, Barski A, Cuddapah S, Cui K, Roh TY, Peng W, Zhang MQ, Zhao K. Combinatorial patterns of histone acetylations and methylations in the human genome. Nature genetics. 2008 Jul 1;40(7):897-903.

[45] Costa IG, Roider HG, do Rego TG, de Carvalho FD. Predicting gene expression in T cell differentiation from histone modifications and transcription factor binding affinities by linear mixture models. BMC bioinformatics. 2011 Feb 15;12(1):S29.

[46] Celniker SE, Dillon LA, Gerstein MB, Gunsalus KC, Henikoff S, Karpen GH, Kellis M, Lai EC, Lieb JD, MacAlpine DM, Micklem G. Unlocking the secrets of the genome. Nature. 2009 Jun 18;459(7249):927-30.

[47] Harrow J, Frankish A, Gonzalez JM, Tapanari E, Diekhans M, Kokocinski F, Aken BL, Barrell D, Zadissa A, Searle S, Barnes I. GENCODE: the reference human genome annotation for The ENCODE Project. Genome research. 2012 Sep 1;22(9):1760-74.

[48] Quinlan AR, Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics. 2010 Mar 15;26(6):841-2.

[49] Ciregan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification. InComputer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on 2012 Jun 16 (pp. 3642-3649). IEEE.

[50] Gong Y, Wang L, Guo R, Lazebnik S. Multi-scale orderless pooling of deep convolutional activation features. InEuropean conference on computer vision 2014 Sep 6 (pp. 392-407). Springer International Publishing.

[51] Taigman Y, Yang M, Ranzato MA, Wolf L. Deepface: Closing the gap to human-level performance in face verification. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2014 (pp. 1701-1708).

[52] Sun Y, Wang X, Tang X. Deep learning face representation from predicting 10,000 classes. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2014 (pp. 1891-1898).

[53] Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229. 2013 Dec 21.

[54] Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. InProceedings of the IEEE conference on computer vision and pattern recognition 2014 (pp. 580-587).

[55] Ouyang W, Luo P, Zeng X, Qiu S, Tian Y, Li H, Yang S, Wang Z, Xiong Y, Qian C, Zhu Z. Deepid-net: multi-stage and deformable deep convolutional neural networks for object detection. arXiv preprint arXiv:1409.3505. 2014 Sep 11.

[56] Parascandolo G, Huttunen H, Virtanen T. Recurrent neural networks for polyphonic sound event detection in real life recordings. InAcoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on 2016 Mar 20 (pp. 6440-6444). IEEE.

[57] Sainath TN, Vinyals O, Senior A, Sak H. Convolutional, long short-term memory, fully connected deep neural networks. InAcoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on 2015 Apr 19 (pp. 4580-4584). IEEE.

[58] Zuo Z, Shuai B, Wang G, Liu X, Wang X, Wang B, Chen Y. Convolutional recurrent neural networks: Learning spatial dependencies for image representation. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops 2015 (pp. 18-26).

[59] Cakir E, Parascandolo G, Heittola T, Huttunen H, Virtanen T. Convolutional recurrent neural networks for polyphonic sound event detection. arXiv preprint arXiv:1702.06286. 2017 Feb 21.

[60] Fawcett T. An introduction to ROC analysis. Pattern recognition letters. 2006 Jun 30;27(8):861-74.

[61] Hajian-Tilaki K. Receiver operating characteristic (ROC) curve analysis for medical diagnostic test evaluation. Caspian journal of internal medicine. 2013;4(2):627.

[62] Powers DM. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011.

[63] Hand DJ. Measuring classifier performance: a coherent alternative to the area under the ROC curve. Machine learning. 2009 Oct 1;77(1):103-23.

[64] Gagn RM. Presidential address of division 15 learning hierarchies. Educational psychologist. 1968 Nov 1;6(1):1-9.